

THE FUTURE OF DIGITAL TERRAIN  
IN  
DISTRIBUTED SIMULATIONS

Captain Rodney A. Houser

Joint Advanced Distributed Simulation Joint Test Force  
11104 Menaul Blvd NE  
Albuquerque, New Mexico 87112

**DISTRIBUTION STATEMENT A**  
Approved for Public Release  
Distribution Unlimited

November 10, 1998

20000619 098

## Contents

1.0 Background.....	1
2.0 Introduction .....	2
3.0 Methodology .....	2
3.0.1 Designing the Advanced Radar Imaging and Emulation System .....	3
3.0.2 Getting the Data .....	3
3.0.2.1 National Imagery and Mapping Agency Digital Data.....	4
3.0.2.2 Digitizing .....	5
3.0.3 Putting the Data into a Standard Form .....	5
3.0.3.1 Datums.....	5
3.0.3.2 Map Projections.....	5
3.0.3.3 Map Scales .....	5
3.0.4 Manipulating the Data .....	6
3.0.5 Limitations .....	11
4.0 Lessons Learned.....	12
5.0 Recommendations .....	13
6.0 Conclusion .....	14

## Appendices

Appendix A - Feature Identification Codes supported by ARIES .....	15
Appendix B - Modified DFAD Feature Attributes .....	17
Appendix C - Digital Data Base ICD for ARIES .....	18
Appendix D - ARC/INFO® Macro Language Scripts.....	24
Appendix E - C Code .....	25
Appendix F - FORTRAN Code .....	27
Appendix G - PV-WAVE® Procedures.....	31
Appendix H - Glossary.....	39

## List of Figures

Figure 1 ARIES System Design .....	3
Figure 2 Area of Interest - Southwest Asia .....	4
Figure 3 Converting between ARC/INFO® and DFAD .....	6
Figure 4 Relationship between DFAD and ARC/INFO® Attributes.....	7
Figure 5 Terrain Database Development Process.....	10
Figure 6 Converting to the ARIES format .....	11

## 1.0 Background

The key objective of the Joint Advanced Distributed Simulation Joint Test Force (JADS JTF) is to provide the Test and Evaluation (T&E) community with an evaluation of the utility of Advanced Distributed Simulation (ADS) as a methodology. The End To End (ETE) test evaluates the utility of ADS to complement the Developmental Test and Evaluation (DT&E) and Operational Test and Evaluation (OT&E) of a Command, Control, Communications, Computers, Intelligence, Surveillance & Reconnaissance (C<sup>4</sup>ISR) system. The test uses the critical operational issues for the Joint Surveillance Target Attack Radar System (Joint STARS) to conduct its T&E utility evaluation in an ADS-enhanced test environment.

The ETE test consists of four phases. The first two phases occur in a laboratory environment, suited for exploring DT&E and early OT&E applications. Phase 3 checks compatibility of the ADS environment with the actual Joint STARS equipment, and Phase 4 is a live open-air test designed to mix live and virtual targets and provide an end-to-end environment for testing Joint STARS in its operational environment. The intent is to provide a set of interfaces from sensor to weapon system including some of the intermediate nodes that would be found in a tactical engagement. The test traces a thread of the battlefield process, from target detection to target assignment, target engagement, and battle damage assessment at corps level, using ADS. It allows the tester to evaluate the thread as a whole and to evaluate what effects an operationally realistic environment has on the system under test. The ETE test is designed to add additional entities in a seamless manner to the battlefield seen by Joint STARS. In addition, the ETE test adds, via ADS, some of the complimentary suite of the C<sup>4</sup>ISR systems and weapons systems with which Joint STARS interacts. This enables the test team to evaluate the utility of an ADS-enhanced test environment.

The ETE test uses ADS, as defined by IEEE Standard 1278 for Distributed Interactive Simulation (DIS), to supplement the operational environment that E-8C and Light Ground Station Module (LGSM) operators would experience. By mixing any available live targets with targets generated by a simulation, the ETE synthetic environment presents a battle array that represents many of the major ground systems found in a corps area of interest. Additionally, by constructing a network with nodes representing appropriate C<sup>4</sup>ISR systems and weapon systems, a more robust cross section of players is available with which the E-8C and LGSM operators can interact.

Several components are required to create the ADS-enhanced operational environment (ETE synthetic environment) that is used in the ETE test. In addition to Joint STARS, the ETE test requires Janus, a validated simulation capable of generating thousands of entities that represent some of the elements in a threat rear area of operation. Also, simulations of the Joint STARS moving target indicator (MTI) radar and synthetic aperture radar (SAR), collectively called the Virtual Surveillance Target Attack Radar System (VSTARS), are used to insert the simulated entities into the radar stream aboard the E-8C. Other components used to support the test include live elements of the Army's artillery command and control process and the Tactical Army Fire Support Model (TAFSM), which simulates a Battalion of the Army's Advanced Tactical Missile System. These simulations begin to interact after an operator starts a scenario in the DIS version of Janus. As VSTARS processes the simulated entities, the LGSM receives MTI and can request

SAR images. Using doctrinally correct means, a soldier sends free text messages from the Compartmented All-Source Analysis System Message Processing System to two remote workstations (RWSs). In turn, a soldier at an Advanced Field Artillery Tactical Data System (AFATDS) receives Target Intelligence Data messages from the RWSs. The AFATDS operator sends a fire mission to another AFATDS operating as a Battalion Fire Direction Center. Here, TAFSM encapsulates the fire and detonation traffic within DIS protocol data units (PDUs) and broadcasts the PDUs across the ETE synthetic environment. Finally, Janus receives the PDUs, assesses damage, and continues the end to end loop.

## **2.0 Introduction**

With the rapidly changing environment of distributed testing, one thing must stay constant—digital terrain. In September 1996, the ETE test team realized that the simulations being developed to support ADS needed to work from the same terrain database. In order to reduce the level of effort required to develop digital terrain for the ETE test, the team used the relatively featureless terrain of Southwest Asia (SWA).

Three of the four simulations (Janus, TAFSM, and the VSTARS MTI) in the ETE synthetic environment already used National Imagery and Mapping Agency (NIMA) digital data. Therefore, the ETE team decided to use this terrain data as a basis. The team contracted Lockheed Martin Tactical Defense Systems (LMTDS) of Litchfield Park, Arizona to design and build the fourth simulation, a Joint STARS SAR simulation called the Advanced Radar Imaging Emulation System (ARIES).

From a distributed testing standpoint, it was essential that each simulation represent terrain and features accurately. Therefore, all four simulations used terrain data derived from Level 1 Digital Terrain Elevation Data (DTED) and Digital Feature Analysis Data (DFAD). Despite using a common terrain basis, approximately 90% of the development effort was correcting the data, adding detail, and putting it into a standard format. This was accomplished by combining the strengths of Geographic Information System (GIS) software tools such as ArcView® and ARC/INFO® with the flexibility of PV-WAVE®, and C and FORTRAN code.

The process and standards involved in producing terrain databases needs to change to make it less time and manpower intensive. This report focuses on the steps used to develop the ETE terrain database, lessons learned, and what can be done to improve the whole process in the future.

## **3.0 Methodology**

The ETE test team based its testing on a 54-hour Corps Battle Simulation scenario in SWA, used by TEXCOM Lab for testing C<sup>4</sup>I systems. This scenario was adapted from the *US Army Command and General Staff College (CGSC) Common Teaching Scenario - Southwest Asia*, dated April, 1992, modified by Headquarters, TRADOC. Not only did the scenario dictate the entities present in Janus, but it also defined SWA as the area of interest for the ETE terrain database. Therefore, in order to interact in the ETE synthetic environment, Janus, TAFSM, and

VSTARS required a terrain database of SWA. The following steps highlight the major concepts used in developing the ETE terrain database.

### 3.0.1 Designing the Advanced Radar Imaging and Emulation System

ARIES, a component of VSTARS, ultimately defined how the ETE terrain database would be built. During the ARIES design process, two implementations were considered—a point map and raster ground truth generation. Figure 1 represents an overview of the final ARIES system design. As shown below, the point map design was selected. At the time, this design offered distinct advantages. First, DTED and DFAD data was readily available. Second, the tools to modify and create new terrain databases were available. Finally, all of the simulations in the ETE synthetic environment could derive their own proprietary formats from the single terrain database. Refer to Appendix C for the interface control document (ICD), which defines the characteristics and structure of the digital terrain elevation and feature database for the ARIES simulation.

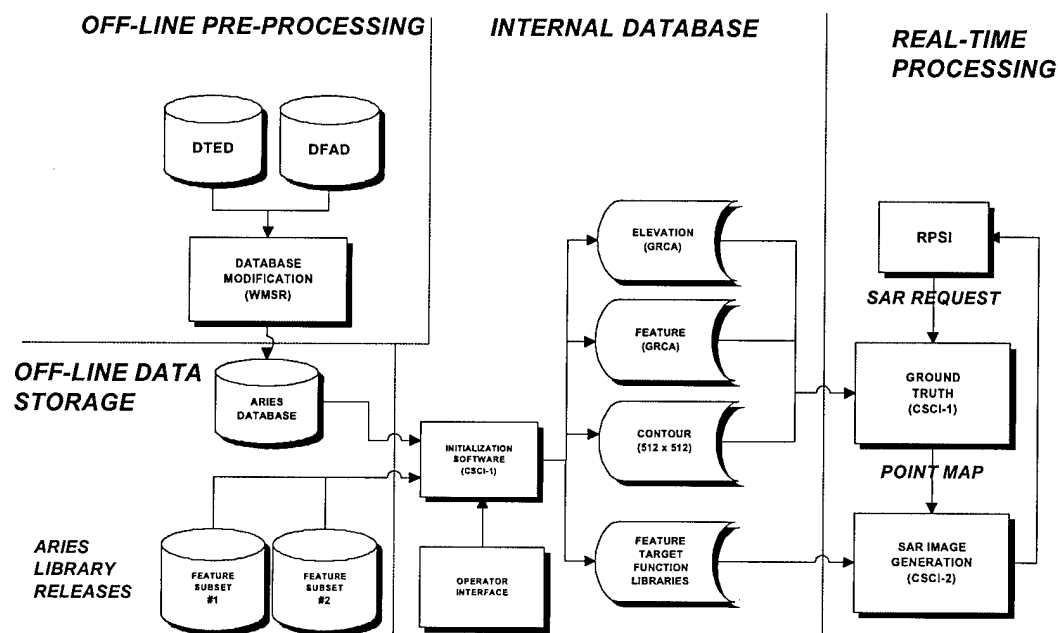


Figure 1 ARIES System Design

### 3.0.2 Getting the Data

The ETE team obtained terrain data as paper maps and in digital format from NIMA. The *Catalog of Maps, Charts, and Related Products* and the *Semiannual Bulletin Digest* proved to be invaluable for determining availability of terrain data for SWA. Figure 2, from TEXCOM's *The Road to War*, shows the area of interest for ETE terrain database development.

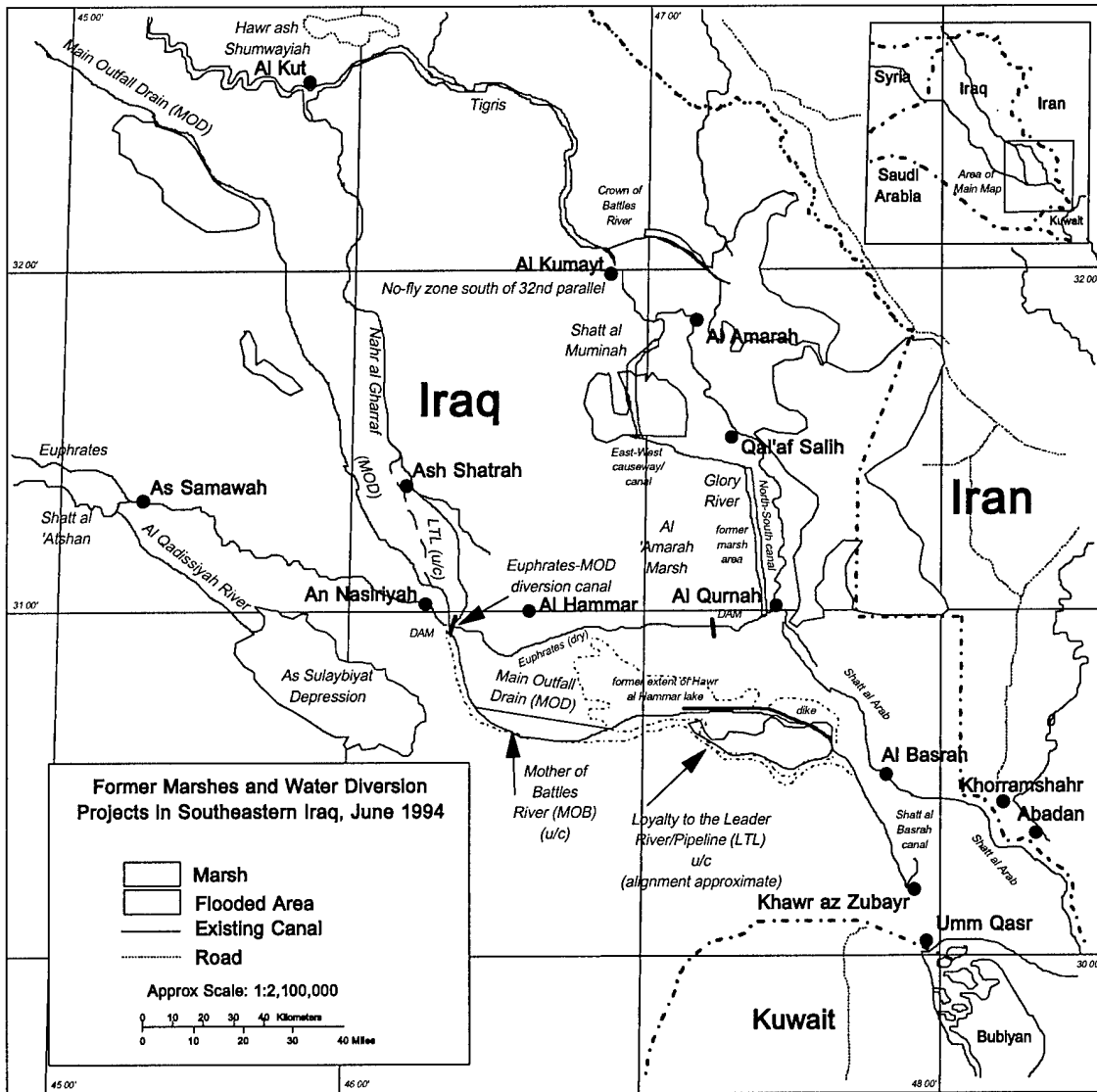


Figure 2 Area of Interest - Southwest Asia

### **3.0.2.1 National Imagery and Mapping Agency Digital Data**

Level 1 DTED and DFAD can be obtained on CD-ROM from the National Imagery and Mapping Agency, 3200 S. Second St., St. Louis, MO 63118. The request should indicate full one-degree geographic cells of data by either southwest corner coordinates, or by delineating the required area.

### **3.0.2.2 Digitizing**

Since the Level 1 DFAD did not provide sufficient detail for ARIES, TEXCOM labs obtained medium scale paper maps (e.g., 1:50,000 and 1:100,000) from the NIMA. The ETE test team digitized features from these maps to augment Level 1 DFAD.

## **3.0.3 Putting the Data into a Standard Form**

When pulling the data together, it was unlikely that it would all be in a consistent format. Therefore, three important references were considered during ETE terrain database development—datums, map projections, and map scales.

### **3.0.3.1 Datums**

In general terms, the size and shape of the Earth is modeled as a spheroid. A geodetic datum uses this approximation to define the mathematical relationship of the size and shape of the Earth to a coordinate system. Since there are many methods to describe the Earth, different countries and agencies use different datums to identify coordinates in GIS software. Referencing coordinates to the wrong datum can result in position errors of several kilometers, so it is important to be aware of the variety of datums. Coordinates in the ETE terrain database are in World Geodetic System 1984 (WGS84).

### **3.0.3.2 Map Projections**

A map projection portrays the surface of the Earth on a 2-D plane. Unfortunately, projection always creates distortions of the Earth's surface in shape, scale, and area. The key to selecting the best projection is determining which projection minimizes those distortions most important to the cartographer. DFAD is distributed in geographic coordinates (latitude and longitude). All of the paper maps used in developing the ETE terrain database were in the Universal Transverse Mercator (UTM) projection. In order to merge the two in GIS software, UTM coordinates were transformed into geographical coordinates. Later, the coordinates were projected into the Topocentric Coordinate System (TCS) for use in VSTARS.

### **3.0.3.3 Map Scales**

The map scale (e.g., 1:1,000,000 or 1:50,000) is the relationship between the distance on a paper map to the same distance on the Earth's surface. Mathematically, this relationship is:

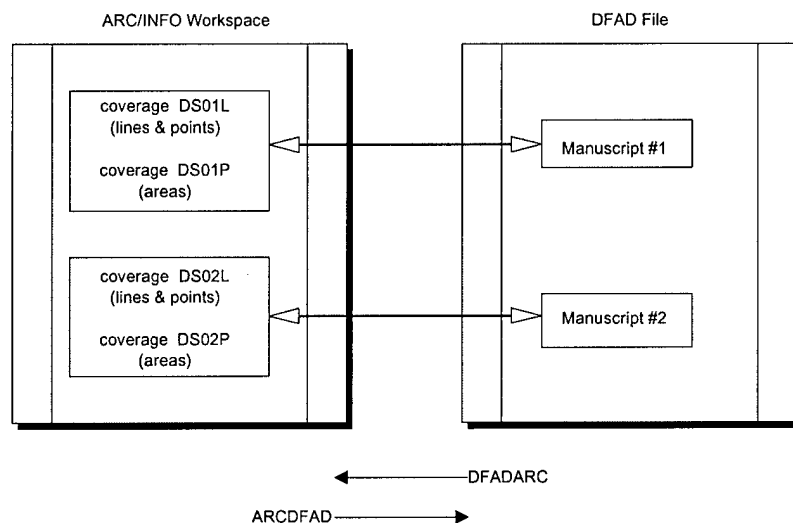
$$\frac{1}{(\text{Scale Denominator})} = \frac{(\text{Map Distance}) \times (\text{Units Conversion})}{(\text{Earth Distance})}$$

The map scale also determines how features are depicted on a paper map. Feature representation changes with map scale. A map scale with a small, scale denominator has the greatest feature detail and is ironically considered to be a large-scale map. With a small-scale map, the location and size and shape of features become distorted. Some features are even omitted. Therefore, it is best to work with the largest map scale that is available.

### 3.0.4 Manipulating the Data

The first question in developing a digital terrain database is “How large of an area is the database going to cover?” This is an important question in that the answer determines how the terrain database is built. A second, and equally important question, is “What is the location of the database?” An area with a number of features also affects how the terrain database is built. In either case, the idea is to convert individual DFAD cells into the largest area possible.

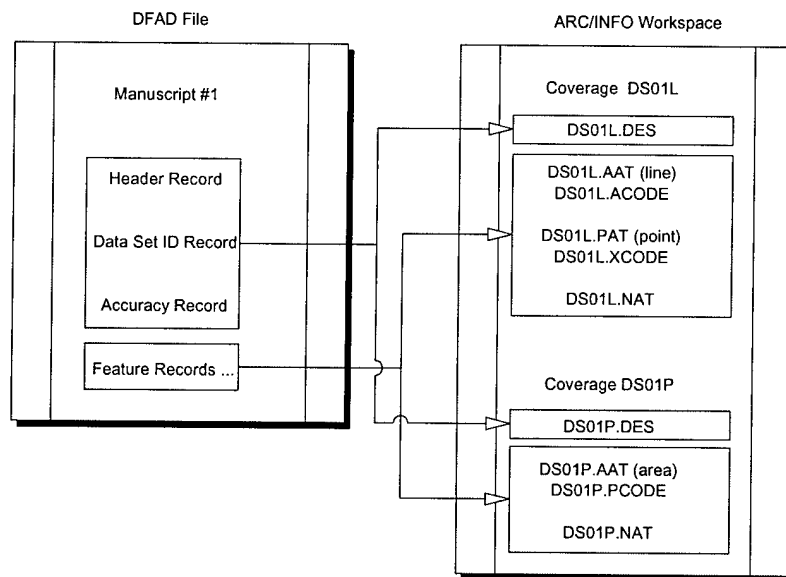
A DFAD file consists of a set of manuscripts that contain point, line, and area features over a 1° by 1° geographic cell. Each manuscript is a database of geographic coordinates and attributes that identify natural and man-made features to a specific level of accuracy. See MIL-PRF-89005 for the DFAD performance specification. When ARC/INFO® converts a DFAD file, it creates a workspace with two coverages per manuscript (DS01P...DS0nP and DS01L...DS0nL), where *n* corresponds to the number of manuscripts in the DFAD file. See Figure 3 for the graphical relationship between an ARC/INFO® workspace and DFAD file.



**Figure 3 Converting between ARC/INFO® and DFAD**



Each DS0nL coverage contains both line and point features. Each DS0nP coverage contains area features. Furthermore, ARC/INFO® coordinate and attribute files are related by an item called FACODE. The FACODE in the DS0nL.ACODE, DS0nL.XCODE, and DS0nP.PCODE files are related to the DS0nL.AAT, DS0nL.PAT, and DS0nP.AAT cover\_IDs respectively. Figure 4 illustrates how point, line, and area features and attributes are stored in ARC/INFO®.



**Figure 4 Relationship between DFAD and ARC/INFO® Attributes**

The following paragraphs give a technical “how-to” create digital terrain databases. Figure 5 illustrates the ETE terrain database development process. Of particular note, the “edgematch & attribute check” and “digitize” steps require the greatest resources. Although this is the method used to create the ETE terrain database, there may be a better process. The recommendations section in this report explores other techniques that may be used in the future. Since the terrain database was created using ARC/INFO®, software specific command line entries are bolded with square brackets. These steps also assume correct references to datum, projection, and scale. Finally, the source code for any references to ARC/INFO® macro language (AML) scripts, PV-WAVE® procedures, and C and FORTRAN code can be found in the appendices of this report.

#### ☛ **Run DFADARC.**

**[DFADARC <dfad file> <workspace>]** This command converts the DFAD cells that will be used in the ETE terrain database into ARC/INFO® coverages and workspaces.

## Join attributes.

**[RUN join\_codes.aml]** This AML joins ARC/INFO® attribute files with the corresponding ARC/INFO® feature file. Refer to Figure 4 to see how files are related.

## Create large coverage (s).

This step creates a large coverage from several ARC/INFO® coverages. A single coverage is created for the area features, and a single coverage is created for the line and point features.

### *Area*

**[PUT <coverage>]** This command is performed for each coverage that will comprise the large coverage. Select the boundary, put it to a new coverage called MASTERBNDS, and then delete the boundary.

**[APPEND <new coverage> line features]** This command joins the area coverages into a large coverage.

**[BUILD <new coverage> line]** This command builds area topology for the new coverage.

**[CLIP < new coverage > <clip box> <clip coverage> line 0.00001]** If a subset of the large coverage is desired, this command will “cookie cut” the large coverage based on the shape of the clip coverage. After using this command, manually “close” any polygons that have been clipped.

**[BUILD <clip coverage> line]** This command builds topology for the new area coverage.

### *Lines & Points*

**[APPEND <new coverage> link features]** This command joins the line and point coverages into a large coverage

**[BUILD <new coverage> line]** This command builds line topology for the new coverage.

**[BUILD <clip coverage> point]** This command builds point topology for the new coverage.

**[CLIP <new coverage> <clip\_box> <clip coverage> link 0.00001]** If a subset of the large coverage is desired, this command will “cookie cut” the large coverage based on the shape of the clip coverage.

**[BUILD <clip coverage> line]** This command builds line topology for the new coverage.

**[BUILD <clip coverage> point]** This command builds point topology for the new coverage.

## ☞ Edgematch & attribute check.

First, manually “edgematch” each new coverage with a back coverage of MASTERBNDS. MASTERBNDS provides a template for all of the original cell boundaries. It is useful because it identifies the areas that need to be edgematched.

Next, correct or create attributes for each feature. See appendix A for a list of supported features in ARIES. Appendix B shows the values used for attributes of features in the ETE terrain database.

Each feature needs a distinct ID. The following commands create separate IDs for all features in the ETE terrain database.

Use these commands for DS0nL.AAT.

```
[CALC ds01l-id = ds01l-id + 10000]
[CALC facode = ds01l-id]
[IDEDIT ds01l line]
```

Use these commands for DS0nL.PAT.

```
[CALC ds01l-id = ds01l-id + 20000]
[CALC facode = ds01l-id]
[IDEDIT ds01l point]
```

Extract the attributes from each coverage as temporary *.acode*, *.pcode*, and *.xcode* files.

[INFODBASE <info file> <dbase file>] This command converts ARC/INFO® files into dbase format.

Use Microsoft® Excel to read the dbase files. Eliminate the duplicate feature entries.

NOTE: The boundary feature should always have an ID = 1.

[DBASEINFO <dbase file> <info file> define] This command converts dbase files back to ARC/INFO® format.

*.acode* Define the ARC/INFO® line files as shown below.

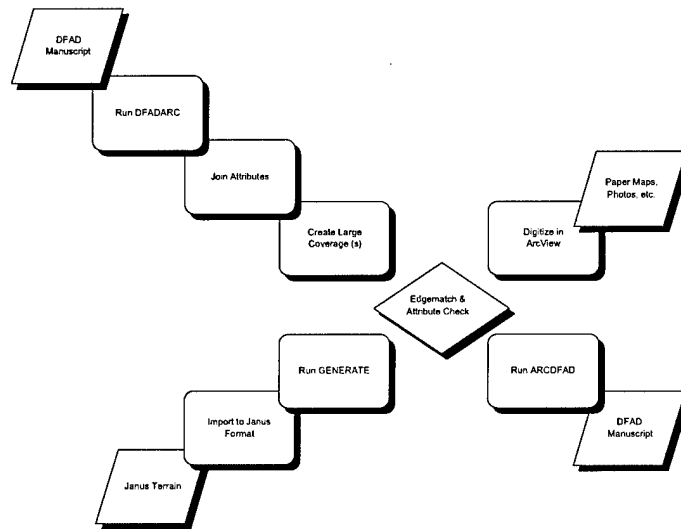
```
facode facode 4 5 B
height height 4 5 B
ficode ficode 4 5 B
smccode smccode 4 5 B
direct direct 1 1 I
width width 2 3 B
```

*.xcode* Define the ARC/INFO® point files as shown below.

```
facode facode 4 5 B
height height 4 5 B
ficode ficode 4 5 B
smccode smccode 4 5 B
orientatio orientation 2 3 B
length length 2 3 B
```

**width width 2 3 B**  
*.pcode* Define the ARC/INFO® area files as shown below.  
**facode facode 4 5 B**  
**height height 4 5 B**  
**ficode ficode 4 5 B**  
**smccode smccode 4 5 B**  
**nstruct\_ps nstruct\_psk 2 5 B**  
**pct\_tree\_c pct\_tree\_cov 2 5 B**  
**pct\_roof\_c pct\_roof\_cov 2 5 B**

**[RUN modify\_codes.aml]** This AML organizes *.pcode*, *.acode*, and *.xcode* files so that features are incremented by 1 in ascending numerical order.



**Figure 5 Terrain Database Development Process**

#### **Creating ARC/INFO® coverages from ArcView® projects.**

This step gets digitized features from ArcView®. After this conversion, the previous step, edgematch & attribute check, must be redone.

**[SHAPEARC <shape file> <coverage>]** This command converts an ArcView® shape file into an ARC/INFO® coverage.

#### **Run ARCDFAD.**

**[ARCDFAD <workspace> <dfad file>]** This command converts ARC/INFO® coverages into a DFAD file.

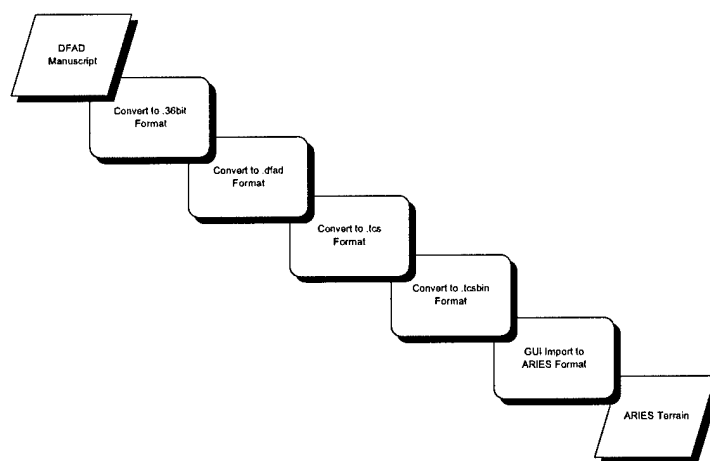
#### **Converting from DFAD to ARIES format.**

***dfad2bits.f*** This FORTRAN program converts 32 bit DFAD to 36 bit words and writes an ASCII file of 36 bit words.

***bits2aries.f*** This FORTRAN program reads the 36 bit ASCII file and writes DFAD features to ARIES format but with coordinates in latitude-longitude.

***aries2tcs.pro*** This PV-WAVE® procedure converts latitude-longitude coordinates into TCS coordinates. The ARIES file format is TCS (X, Y) only.

***tcs2bin.pro*** This PV-WAVE® procedure converts the ARIES file to binary format.



**Figure 6 Converting to the ARIES format**

#### **Creating Janus terrain from ARC/INFO® coverages.**

[**UNGENERATE** <line | point> <coverage> <text file>] This command converts ARC/INFO® coverages into (x, y) point files.

***janus2arc.c*** This C program converts (x, y) point files into a format that can be read by Janus terrain input tools.

### **3.0.5 Limitations**

The greatest limitation of creating the ETE terrain database is the quality of the data. Although most of the digital data is obtained from NIMA, there is no consistency between geographic cells of data. Often, features shown on a comparably scaled paper map do not match, or worse, are missing. This brings up an interesting dilemma. “Do you use DFAD Level 1 in its original format?” In the case of the ETE terrain database, more detailed digital data was not available. “Or, do you spend time and resources to digitize the detail, correct features across geographic cell

boundaries, and modify feature attributes to make them more realistic?" That is what happened with the ETE terrain database.

Only 16,383 features are allowed in each DFAD manuscript. Since the ETE terrain database ultimately contained over 45,000 features, a single manuscript was not possible. Since each manuscript contains lines, points, and area features (refer to figure 4), it made sense to create several smaller manuscripts to define the area. The basic premise was to divide manuscripts based on existing DFAD geographic cell boundaries. This organized the area and made it easier to edgematch. Dividing cells independently of existing boundaries, while possible, would have added unnecessary complexity and manhours to the project. The benefit of this approach was that it was possible to set up separate manuscripts to distinguish between original DFAD and the digitized detail.

The computer hardware (Hewlett-Packard 735 workstation with 144-megabyte memory and 8-gigabyte hard disk storage) used to run ARC/INFO® was adequate for the project; however, using a comparatively inferior workstation could jeopardize a user's ability to quickly navigate (zoom in, out, or across) coverage views. Computer disk space was never a concern. The entire ETE terrain database could have been completed using a 4-gigabyte hard disk drive. A Pentium® class personal computer and digitizer is required for digitizing into ArcView® and performing other data manipulations. Finally, the computers need to be networked to facilitate the exchange of data between ArcView®, ARC/INFO®, and Microsoft® Excel.

Finally, during the ETE terrain database development, there was no ability to view features as they would appear in ARIES. Feature attributes (length, width, height), as defined by the DFAD specification, were coded in the database as half their actual value and then only as a whole number. Also, the feature orientation was coded in the database as a whole number from zero to eight, with each number representing the orientation of the feature's length as a multiple of 11.25 degrees from true north. Refer to appendices A and B for the following example. A single family dwelling (coded as length = 6, width = 4, height = 2, and orientation = 4) would be a 12-meter by 8-meter by 4-meter feature oriented at 45 degrees in ARIES. Imagine thousands of features in any given area. It would be nice to know how these attributes affect one another with a 3-D visual representation.

#### **4.0 Lessons Learned**

In order to accomplish the project, it is essential to not only have the software, but have personnel trained to use it. It took approximately 18 months to develop the ETE terrain database. Four of those months were spent learning the GIS software and developing the processes that would be used to create the database. An individual with GIS software experience could easily shave 25% off this timeline, if not more. Adding additional people would also decrease development time. In this case, resources would be best allocated during the laborious edgematching and digitizing steps shown in figure 5.

The level of effort required to develop a terrain database is time and manpower intensive, but careful planning can shorten the development schedule. Time can be saved by not having to correct the digital data.

- Get the most recent and accurate data available from NIMA.
- Use a consistent datum, for example WGS84, when digitizing from paper maps.
- Determine what fidelity is required for the terrain database.

The ETE terrain database was developed for the relatively featureless desert of SWA. The additional terrain and feature complexity of Bosnia or Korea would require a GIS team and systematic plan to divide and conquer the terrain database.

Map error is cumulative and comes from various sources. Using map projections introduces error. Carelessly mixing datums introduces error. Digitizing introduces error. Converting between file formats of different terrain databases introduces error. The only way to minimize map error in distributed simulations is to ensure that all simulations use a common terrain database.

## **5.0 Recommendations**

There are several emerging technologies that could improve ETE terrain database development in the future. First, an easy method to transfer features from intelligence sources (such as satellite photos) into a digital format would provide the capability to quickly add exceptional detail to terrain databases. R2V™ from Able Software Company, ERDAS® IMAGINE Advantage™ with IMAGINE vector module, and AUTOGRAPHICS® from LMTDS, Akron, OH are three software packages that would facilitate raster to vector conversion during terrain database development. Second, eliminating conversions between file formats of different terrain databases would minimize map errors among distributed simulations. Vector Product Format (VPF) is meta-data and the Synthetic Environment Data Representation & Interchange Specification (SEDRIS) is a meta-model that promises to be compatible with a wide variety of applications.

R2V™ is a simple, intuitive software package that automatically vectorizes raster images. R2V™ has standard vector editing tools, but its sub-par image processing tools would limit projects with complex images. R2V™ can label, georeference, and export vector data to other major GIS formats. Another great feature is the ability to merge multiple vector files. For more information, visit Able Software Company on the web at <http://www.ablesw.com>.

The ERDAS IMAGINE® software package is more robust than R2V™. ERDAS IMAGINE® works with a variety of raster and vector formats. ERDAS® IMAGINE Advantage™ allows the user to directly access image data in native format, and then display and link multiple data files. When combined with the vector module, this important feature gives the user the ability to create and edit vector data from images. Orthorectification, advanced image processing, and spatial analysis make it easier to develop accurate databases. For more information, visit ERDAS® on the web at <http://www.erdas.com>.

The AUTOGRAPHICS® software package allows the user to “train” the software to extract features from a raster image of a paper map. First, the user selects an example of each feature from the raster image with point-and-click actions. After the user identifies examples, the software automatically classifies the remaining features in the image. For more information, contact LMTDS Business Development in Akron, Ohio at (330) 796-4747.

MIL-STD-2407 defines the VPF standard. VPF data, also called meta-data, is arranged as directories, tables and indices. Essentially, VPF provides a model that describes the structure, organization, and relationships of the information in a terrain database. This allows fast, direct access of the database with no need for translation. The NIMA web site at <http://www.nima.mil> has more information about VPF.

SEDRIS uses the concept of meta-data and extends it to the synthetic environment. SEDRIS calls for an application programmer interface (API) to access a terrain database. An API converts between a simulation’s native data format and the SEDRIS model. This means that simulations in a synthetic environment can truly be interoperable. Instead of relying on custom terrain databases, each simulation in the synthetic environment could simply use an API to interact with a common terrain database. Another distinct advantage of the SEDRIS model is that it is easier to communicate with other simulations by describing the data through attributes than through a data storage format. For more information, visit the SEDRIS home page on the web at <http://www.sedris.net>.

## **6.0 Conclusion**

Terrain database development is labor intensive and time-consuming. However, a GIS manager can organize a tool chest of software that makes building terrain databases easier. Careful planning and ample resource allocation ensures that the terrain database is completed quickly. Once the terrain database has been developed, have all the simulations in the synthetic environment use a common terrain database. By eliminating the need for terrain database conversion between simulations, you speed up data access and minimize the map errors that inevitably plague distributed simulations.



## Appendix A - Feature Identification Codes supported by ARIES

The following Feature Identification (FID) codes are used to describe the predominant nature of all features (area, linear, and point) that are supported by ARIES.

<u>Feature Identification</u>	<u>FID Code</u>
<u>Area Features</u>	
Quarry .....	102
Depot.....	778
Soil.....	902
Packed Sand & Gravel.....	906
Sand Dunes.....	907
Salt Marsh.....	908
Smooth Solid Rock.....	910
Rocky Flat .....	912
Dry Lake.....	913
Flood Plain.....	914
Loose Sand.....	917
Dry Depression.....	918
Wadi .....	919
Salt Flat.....	934
Fresh Water (General).....	940
Non-Perennial Stream (Linear Portrayal).....	945
Orchard/Hedgerow (Background).....	951
Irrigated Field.....	958
<u>Linear Features</u>	
Railroad.....	206
Dual Highway (with Median).....	250
All Weather Hard Surface Highway.....	251
All Weather Loose or Light Surface Road.....	252
Fair Weather Loose or Light Surface Road.....	253
Cart Track, Trail.....	254
Road, Approximate Alignment, Under Construction, Existence Reported.....	255
Pipeline (Above Ground).....	281
Powerline Pylon (Type "A").....	541
Powerline Pylon (Type "H").....	542
Powerline Pylon (Type "I").....	543
Powerline Pylon (Type "Y").....	544
Runway and Taxiway.....	706
Cleared Way.....	916
Wadi .....	919
Levee.....	921
Wall.....	922
Escarpment.....	924
Chain Link Fence.....	927
Fresh Water.....	940
Non-Perennial Stream (Linear Portrayal).....	945
Canal/Channelized Stream/Drainage Ditch, (Subject to Ice, Linear Portrayal).....	947
Revetment.....	981
Berm .....	982

Barbed Wire Fence.....	983
Concertina Fence.....	984
Ditch.....	985
Trench.....	986

#### Point Features

Gas/Oil Derrick.....	103
Offshore Platform.....	104
Refinery.....	120
Power Plant (General).....	130
Substation.....	138
Light Fabrication Industry (General).....	160
Associated Structure (General Industry).....	180
Building.....	181
Smokestack.....	182
Pumping Station.....	184
Railroad Station.....	222
Bridges (General).....	260
Associated Structure (General Transportation).....	290
Commercial Building (General).....	301
Grandstand.....	324
Multi-Family Dwelling (General).....	401
Single Family Dwelling (General).....	420
Agricultural Building (General).....	430
Cemetery Building.....	451
Communication Tower.....	501
Miscellaneous Tower.....	530
Power Transmission Tower (General).....	540
Governmental (General).....	601
Prison.....	604
School.....	620
Hospital.....	630
House of Religious Worship (General).....	650
Associated Structure (General Institutional).....	680
Airport/Airbase (General).....	701
Ground Support Facility (General).....	770
Tank (General).....	801
Grain Elevator.....	822
Water Tower (Building).....	824
Warehouse.....	861
Date Palm.....	957

## Appendix B - Modified DFAD Feature Attributes

FID	HEIGHT	SMCCODE	ORIENTATION	DIRECT	LENGTH	WIDTH
	(m)		(deg)		(m)	(m)
251	0	14	-	3	-	4
252	0	14	-	3	-	4
253	0	5	-	3	-	4
254	0	5	-	3	-	2
281	2	2	-	2	-	1
706	0	9	-	3	-	25/12/8/4
921	3/1	5	-	2	-	6/3
922	3	3	-	2	-	2
927	3	1	-	2	-	1
940	0	6	-	2	-	6
947	0	6	-	2	-	15
984	2	1	-	2	-	1
103	15	2	0 - 8	-	25	25
130	15	2	0 - 8	-	25	25
138	5	3	0 - 8	-	15	15
180	5	3	0 - 8	-	15	15
181	5	3	0 - 8	-	15	15
182	15	3	0 - 8	-	2	0
184	5	3	0 - 8	-	15	15
222	4	2	0 - 8	-	25	25
420	3/2	3	0 - 8	-	10/6	4
430	5	3	0 - 8	-	15	15
601	5	3	0 - 8	-	15	15
650	5	3	0 - 8	-	15	10
701	3	3	0 - 8	-	12/5	12/5
770	2	3	0 - 8	-	10/2	8/2
801	4	1	0 - 8	-	4	0
824	6	1	0 - 8	-	2	0
861	5	3	0 - 8	-	25	25
957	4	12	0 - 8	-	2	0

DIGITAL DATA BASE  
Interface Control Document  
for the  
ADVANCED RADAR IMAGING  
EMULATION SYSTEM (ARIES)

Contract Number: F33615-95-C-1610

CDRL Sequence Number: A025

30 September 1996

Prepared for:

Air Force Wright Laboratory

Prepared by:

Lockheed Martin Tactical Defense Systems  
Post Office Box 85  
Litchfield Park, Arizona 85340-0085

This document includes proprietary data that shall not be disclosed outside Lockheed Martin Tactical Defense Systems and its designated parties and shall not be duplicated, used, or disclosed, in whole, or in part, for any purpose.

This restriction does not limit the customer's right to use information contained in the data if it is obtained from another source without restriction.

## TABLE OF CONTENTS

1. SCOPE .....	19
1.1 Purpose .....	20
1.2 Application .....	20
1.3 Definitions and Conventions .....	20
2. APPLICABLE DOCUMENTS.....	20
3. DATA BASE DESCRIPTIONS .....	20
3.1 Data Base Coordinate System.....	20
3.1.1 UNITS OF MEASURE.....	20
3.1.2 HORIZONTAL REFERENCE.....	20
3.1.3 VERTICAL REFERENCE .....	20
3.1.4 MAXIMUM COORDINATES.....	21
3.2 Digital Terrain Data Base.....	21
3.2.1 TERRAIN DATABASE STRUCTURE .....	21
3.3 Digital Feature Data Base.....	21
3.3.1 FEATURE DATA BASE STRUCTURE.....	22
4.3 Digital Contour Terrain Data Base.....	22
4.3.1 CONTOUR TERRAIN DATA BASE STRUCTURE .....	22

## **1. SCOPE**

This document defines the contents of the digital terrain and feature data base to be used by the ARIES Synthetic Aperture Radar (SAR) Imagery Simulation being developed by Lockheed-Martin Tactical Defense Systems (Lockheed-Martin) for incorporation into the Radar Processor Simulation (RPS) on the Joint STARS platform. The RPS is being developed for the integration of Joint STARS into the Joint Advanced Distributed Simulation (JADS) environment.

### **1.1 Purpose**

The purpose of this document is to define the characteristics and structure of the digital terrain elevation and feature data base for the ARIES simulation. The SAR image produced by the ARIES simulation represents the terrain elevation characteristics and the specific features and their locations in the area simulated. This data base will be structured Defense Mapping Agency Digital Terrain Elevation Data and Digital Feature Analysis Data products.

### **1.2 Application**

Interface requirements set forth in this document apply during the development and testing of the ARIES SAR simulation and the RPS.

### **1.3 Definitions and Conventions**

The following conventions were used to describe each message interface:

## **2. Applicable Documents**

The following documents are applicable to the extent specified herein:

DMA Digital Terrain Elevation Data (DTED) Specification

DMA Digital Feature Analysis Data (DFAD) Specification

## **3. Data base descriptions**

### **3.1 Data Base Coordinate System**

The data base shall utilize a topocentric coordinate system. The coordinate system uses a reference plane tangent to the earth at the Latitude and Longitude specified when the data base is constructed. Lines in the reference plane are orthogonal while lines of Longitude on the earth's surface curve together as Latitude increases.

#### **3.1.1 Units of Measure**

All measurements shall be in meters.

#### **3.1.2 Horizontal Reference**

The axes in the plane shall be oriented East-West (X) and North-South (Y). Displacements to the North and East from the topocentric center shall be positive. Displacements to the South and West shall be negative.

#### **3.1.3 Vertical Reference**

Elevation at the data points shall be referenced to the topocentric plane. Points with elevations below the plane shall be negative, points above the plane shall be positive. Points which fall in the plane will have an elevation of zero. It should be noted that points of constant elevation referenced to Sea Level in the DTED data base will produce elevations that vary in the topocentric coordinate system, based on the distance from the JSTARS topocentric center.

### 3.1.4 Maximum Coordinates

The maximum displacement along either horizontal axis from the topocentric center shall be  $\pm 256,000$  meters. The maximum displacement along the vertical axis shall be  $-12,000$  meters to  $+9,000$  meters.

### 3.2 Digital Terrain Data Base

The ARIES digital terrain data base will be derived from DMA DTED Level 1 terrain elevation data. The terrain data base will merge data from a number of standard DMA DTED files to produce one data base covering the entire 512 KM x 512 KM area.

A coordinate conversion will be required to map the DTED data referenced to a Latitude-Longitude coordinate system to the JSTARS topocentric coordinate system.

#### 3.2.1 Terrain Database Structure

The terrain database shall be structured in the same manner as specified in the DTED specification. The number of entries at any given latitude will be a constant due to the orthogonality of the coordinate system compared to the Latitude-Longitude coordinate system.

### 3.3 Digital Feature Data Base

The ARIES digital feature data base will be derived from DMA DFAD Level 1 feature data. The feature data base will merge data from a number of standard DMA DFAD files to produce one data base covering the entire 512 KM x 512 KM area.

A coordinate conversion will be required to map the DFAD data referenced to a Latitude-Longitude coordinate system to the JSTARS topocentric coordinate system.

Additional features may be added to the data base at the direction of the JADS program office. Format for these features shall be in accordance with the DMA DFAD specification. New types of features will also be added which are not represented by current Feature Identification Numbers (FID).

The table below defines the new features (non-DFAD) to be added to the Digital Feature Database.

#### Non-DFAD Feature Definitions

<u>NAME</u>	<u>FID</u>	<u>TYPE</u>
Chain Link Fence	935	linear
Barbed Wire Fence	936	linear
Concertina Fence	937	linear
Anti-Tank Ditches	938	linear
Date Palm Trees	957	point

Some features required are not directly supported by DFAD features but can be indirectly supported using corresponding existing DFAD FID's. The table below identifies these features and the substitute feature to be used.

## **Substitute DFAD Feature Definitions**

<b><u>Non DFAD Feature</u></b>	<b><u>Substitute DFAD Feature</u></b>
Rocky flats	Boulder Field (FID(#911), Rocky, Rough Surface (FID #912)
Packed sand and gravel	Sand/Desert (FID #906)
Loose sand	Sand/Desert (FID #906)
Dry depressions with sandy bottoms	Sand/Desert (FID #906) with DTED
Wadis	Sand/Desert (FID #906) with DTED
Escarments	Ground Surface (FID #902), Sand/Desert (FID #906), Cliffs (FID #924) with DTED
Salt marshes	Ground Surface (FID #902), Marsh/Swamp (FID #908)
Salt flats	Salt Pans (FID #934)
Flood plains	Ground Surface (FID #902), Mud/Tidal Flats (FID #914)
Date palm orchards	Orchards (FID #951)
Irrigated fields	Soil (FID #902), Vegetation (FID #950)
Oil wells	Gas/Oil Derrick (FID #103)
Revetments	Levees/Embankments (FID #921), Low Embankments/Low Levees (FID #980)
Below ground sand/dirt trenches	Ground Surface (FID #902), Sand/Desert (FID #906) with DTED
Sand/dirt ditches	Ground Surface (FID #902), Sand/Desert (FID #906) with DTED
Transmission towers - 4 sided pyramidal	Communications Towers (FID #501), Radio/Television Towers (FID #'s 511, 512), Power Transmission Towers (FID#504)
Electrical power lines	Powerline Pylons (FID #'s 541-544)
Dirt and concrete dikes and levees	Conduits (FID #280), Levees/Embankments (FID #921), Low Embankments/Low Levees (FID #980)
Dirt and concrete walls and berms	Walls (FID #922)
Pipelines within trenches	Pipelines (Above Ground) (FID #281)

### **3.3.1 Feature Data Base Structure**

The feature database shall be structured in the same manner as specified in the DMA DFAD specification. Changes to this structure will occur in the maximum number of features per data set and the limitations on the maximum geographic area covered by the data base.

### **4.3 Digital Contour Terrain Data Base**

The ARIES digital contour terrain data base will be derived from DMA DTED Level 1 terrain elevation data. The contour terrain data base will merge data from a number of standard DMA DTED files to produce one data base covering the entire 512 KM x 512 KM area.

A coordinate conversion will be required to map the DTED data referenced to a Latitude-Longitude coordinate system to the JSTARS topocentric coordinate system.



#### **4.3.1 CONTOUR TERRAIN Data Base Structure**

The contour terrain database shall be structured in contour vectors. Each contour vector shall be separated in elevation by 10 meters. The contour terrain database file will a binary data file as created by PV-WAVE®.

## Appendix D - ARC/INFO® Macro Language Scripts

```

/* startup.aml
/* Unix version of startup.aml
/*
/* Run this aml to set up the Arc/Info environment and initialize
/* convenient variables for each workspace.
/*
/* Created 1/2/97 by Capt Rodney Houser
/*
&term 9999
display 9999 3
coordinate mouse
&setvar .etc = /disk1/users/ai/etc/
&setvar .aries = /disk1/users/ai/etc/aries/
&setvar .janus = /disk1/users/ai/etc/scenario/janus/
precision double
&return

/* join_codes.aml
/* This aml joins the pcode, acode, and xcode files for one coverage. Copy
/* and run this aml from the workspace that contains the coverage.
/*
/* Created 08/07/97 by Capt Rodney Houser
/*
relate restore ds01p.relate
tables
additem ds01p.aat facode 4 5 B # ds01p-id
sel ds01p.aat
calc facode = ds01p-id
q
joinitem ds01p.aat ds01p.pcode ds01p.aat facode facode
regionclass ds01p reg land ds01p-id facode
clean reg reg ## poly
joinitem reg.patland reg.pcode reg.patland facode facode
createlabels reg
relate restore ds011.relate
tables
additem ds011.aat facode 4 5 B # ds011-id
sel ds011.aat
calc facode = ds011-id
q
joinitem ds011.aat ds011.acode ds011.aat facode facode
tables
additem ds011.pat facode 4 5 B # ds011-id
sel ds011.pat
calc facode = ds011-id
q
joinitem ds011.pat ds011.xcode ds011.pat facode facode
&return

/* create_regions.aml
/* This aml creates a new region subclass LAND for DFADARC coverages, and
/* joins attribute information to that subclass. Replace XXXX
/* with the workspace variable name.
/*
/* Created 08/07/97 by Capt Rodney Houser
/*
relate restore ds01pXXXX.relate
tables
select ds01pXXXX.aat
additem ds01pXXXX.aat facode 4 5 B # ds01pXXXX-id
calc facode = ds01pXXXX-id
q
joinitem ds01pXXXX.aat ds01pXXXX.pcode ds01pXXXX.aat facode facode
regionclass ds01pXXXX regXXXX land ds01pXXXX-id facode
clean regXXXX regXXXX ## poly
joinitem regXXXX.patland regXXXX.pcode regXXXX.patland facode facode
createlabels regXXXX
&return

/* modify_codes.aml
/* This aml organizes pcode, acode, and xcode files so that features
/* are incremented by 1 in ascending numerical order. Copy and run
/* this aml from the workspace that contains the final ds01p and
/* ds011 coverages.
/*
/* Created 08/07/97 by Capt Rodney Houser
/*
tables
additem ds011.acode pin 4 5 B # facode
additem ds011.xcode pin 4 5 B # facode

```

```

additem ds01p.pcode pin 4 5 B # facode
/*
sel ds01p.pcode
calc pin = Srecno
sel ds011.acode
calc pin = Srecno
sel ds011.xcode
calc pin = Srecno
sel
/*
dir *code
&pause Identify number of records
&s pcode [response 'Enter number of pcode records']
&s acode [response 'Enter number of acode records']
sel ds011.acode
calc pin = pin + %pcode%
sel ds011.xcode
calc pin = pin + %pcode% + %acode%
q
/*
pullitems ds01p.pcode ds01p.ppin
~facode
~pin
~end
pullitems ds011.acode ds011.apin
~facode
~pin
~end
pullitems ds011.xcode ds011.xpin
~facode
~pin
~end
/*
joinitem ds01p.aat ds01p.ppin ds01p.aat facode facode
joinitem ds011.aat ds011.apin ds011.aat facode facode
joinitem ds011.pat ds011.xpin ds011.pat facode facode
/*
tables
sel ds01p.aat
calc ds01p-id = pin
calc facode = pin
sel ds011.aat
calc ds011-id = pin
calc facode = pin
sel ds011.pat
calc ds011-id = pin
calc facode = pin
sel
/*
dropitem ds011.pat pin
dropitem ds011.aat pin
dropitem ds01p.aat pin
/*
sel ds01p.pcode
calc facode = pin
sel
dropitem ds01p.pcode pin
sel ds011.acode
calc facode = pin
sel
dropitem ds011.acode pin
sel ds011.xcode
calc facode = pin
sel
dropitem ds011.xcode pin
/*
kill ds01p.ppin
kill ds011.apin
kill ds011.xpin
q
/*
idedit ds01p line
idedit ds011 line
idedit ds011 point
&return

```

## Appendix E - C Code

```
#include <sys/file.h>

int iopen (ch, nch)
char *ch;
int *nch;
{
    return (open (ch, O_CREAT | O_WRONLY, 0777));
}

int iopenr (ch, nch)
char *ch;
int *nch;
{
    return (open (ch, O_RDONLY));
}

int iread (fd, buf, nbytes)
int *fd;
char *buf;
int *nbytes;
{
    int num;

    num = read (*fd, buf, *nbytes);
    if (num < 0)
    {
        perror ("nbiniio read");
    }
    return (num);
}

int iwrite (fd, buf, nbytes)
int *fd;
char *buf;
int *nbytes;
{
    int num;

    num = write (*fd, buf, *nbytes);
    if (num < 0)
    {
        perror ("nbiniio write");
    }
    return (num);
}

/* arc2janus.c */

#define PROG_I1 "This program converts an arcinfo ungenerate ascii file"
#define PROG_I2 " (in utm) into janus compatible terrain data input file"

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define NUM_FILETYPES          9
#define TRUE                   1
#define FALSE                  0
#define MAX_NUM_TOKENS        5
#define MAX_TOKEN_CHARS        20
#define NUM_SEPARATOR_CHARS    4
#define MAX_NUM_FEATURES      10000 /* janus limit is 10000 */
#define MAX_NUM_VERTICES      100000 /* janus limit is 100000 */
#define MAX_NUM_VERTICES_PER_FEATURE 1000 /* janus limit is 1000 */

/***** DECLARE FUNCTIONS *****/
void write_feature_set();

int get_tokens (char *string, char *separator, char *token[],
               int max_token_chars, int max_num_tokens);

/***** GLOBAL VARIABLES *****/
int num_features = 0;
int feature_num;
int npts;
double feature_casting [MAX_NUM_VERTICES_PER_FEATURE];
double feature_northing [MAX_NUM_VERTICES_PER_FEATURE];
FILE *fout;
char filetype [256];
```

```
int filetype_id;

/***** MAIN PROGRAM *****/
void main ()
{
    FILE *fin;
    int i;
    int good_file_type;
    char filename [256];
    char outfile [256];
    char filetypes [NUM_FILETYPES][25];

    int num_tokens;
    char line[256];
    char separator[NUM_SEPARATOR_CHARS] = " \t"; /* SPACES OR TABS */
    char *token[MAX_NUM_TOKENS];

    // ALLOCATE STORAGE FOR THE TOKENS
    for (i=0; i<MAX_NUM_TOKENS; i++) {
        if ( (token[i] = (char *) malloc (MAX_TOKEN_CHARS)) == NULL ) {
            fprintf (stderr, "Couldn't allocate token storage\n");
        }
    }

    /* DEFINE THE ACCEPTABLE FILE TYPES */
    strcpy (filetypes[0], "Dual Highway");
    strcpy (filetypes[1], "Highway");
    strcpy (filetypes[2], "Road");
    strcpy (filetypes[3], "River");
    strcpy (filetypes[4], "Runway");
    strcpy (filetypes[5], "Levee");
    strcpy (filetypes[6], "Wall");
    strcpy (filetypes[7], "Fence");
    strcpy (filetypes[8], "Lake");

    /* PROMPT USER FOR INPUT FILE */
    printf ("Enter file name: ");
    printf ("File Selected: %s", filename);

    /* CREATE THE OUTPUT FILENAME */
    strcpy (outfile, filename);
    strcpy (strchr (outfile, '.'), "cnv");

    /* OPEN THE OUTPUT FILE */
    if ( (fout = fopen (outfile, "w")) == NULL ) {
        fprintf (stderr, "ERROR .. couldn't open output file: %s\n",
                outfile);
        exit (0);
    }

    /* PROMPT USER FOR THE TYPE OF FILE */
    printf ("Enter Source file type: ");
    for (i=0; i<NUM_FILETYPES; i++) printf ("%d %s", i, filetypes[i]);

    good_file_type = FALSE;
    while (!good_file_type) {
        printf ("Enter ID of Source file type (99 to exit): ");
        scanf ("%d", &filetype_id);

        if (filetype_id > -1 && filetype_id <= NUM_FILETYPES) {
            good_file_type = TRUE;
        }

        if (filetype_id == 99) {
            printf ("Exiting .. thanks, its been fun!!!\n");
            exit (0);
        }

        if (!good_file_type) {
            printf ("You must enter the ID of one of the Source File types");
            printf (" or 99 to exit !!!");
        }
    }

    printf ("ID of Source file entered: %d Filetype: %s\n",
            filetype_id, filetype);
```

```

        filetype_id, filetype[filetype_id]);

/* OPEN THE INPUT FILE */
if ((fin = fopen(filename, "r")) == NULL) {
    fprintf(stderr, "Unable to open %s\n", filename);
}

/* PERFORM THE CONVERSIONS */
while (fgets(line, sizeof(line), fin) != NULL) {
    *strchr(line, '\n') = '\0'; // REMOVE NEWLINE

    num_tokens = get_tokens(line, separator, token,
        MAX_TOKEN_CHARS, MAX_NUM_TOKENS);

    if (num_tokens == 0) {
        fprintf(stderr, "ERROR .. got zero tokens\n");
        break;
    }

    // IF ONLY 1 TOKEN, IT'S THE END OF A FEATURE SET OR BEGINNING
    if (num_tokens == 1) {
        if (strcmp(token[0], "END") == 0) {
            if (npts == 0) {
                printf("\n\nTHAT'S ALL FOLKS .. new file: %s\n", outfile);
                exit(0);
            } else {
                write_feature_set();
                npts = 0;
            }
        } else {
            feature_num = atoi(token[0]);
            if (feature_num == 0) {
                fprintf(stderr, "ERROR .. got feature number of 0");
                fprintf(stderr, "\n      could be bad data\n");
                exit(0);
            }
        }
    }

    // IF 2 TOKENS, GET THE EASTING, NORTHING FEATURE DATA
    if (num_tokens == 2) {
        feature_easting[npts] = atof(token[0]);
        feature_northing[npts] = atof(token[1]);
        npts++;
        if (npts > MAX_NUM_VERTICES_PER_FEATURE) {
            fprintf(stderr, "ERROR .. exceeded max number vertices\n");
            exit(0);
        }
    }
}

/* END OF MAIN */

/***** WRITE_FEATURE_SET *****/
void write_feature_set()
{
    int i;
    char header_str[MAX_NUM_FEATURES][256];

    strcpy(header_str[0], "6\\1\\1\\16\\Dual Highway");
    strcpy(header_str[1], "6\\2\\2\\8\\Highway");
    strcpy(header_str[2], "6\\3\\3\\8\\Road");
    strcpy(header_str[3], "2\\1\\1\\100\\River");
    strcpy(header_str[4], "3\\1\\1\\8\\Runway");
    strcpy(header_str[5], "3\\2\\2\\12\\Levee");
    strcpy(header_str[6], "3\\3\\3\\1\\Wall");
    strcpy(header_str[7], "4\\1\\1\\0\\Fence");
    strcpy(header_str[8], "5\\1\\1\\0\\Lake");

    printf("\nwriting feature_num: %d with %d vertices", feature_num, npts);

    fprintf(fout, "%d %s %10d\n",
        feature_num, header_str[filetype_id], npts);

    for (i=0; i<npts; i++)
        fprintf(fout, "%10.2f %10.2f\n",
            feature_easting[i], feature_northing[i]);
}

/* END OF WRITE_FEATURE_SET */

```

```

/***** GET_TOKENS *****/
int get_tokens(char *string,
    char *separator,
    char *token[],
    int max_token_chars,
    int max_num_tokens)
{
    /*****
    /* obtains tokens in string */
    /* returns number of tokens found, 0 if unsuccessful */
    /*
    /* the following must be defined prior to calling */
    /* token: pointer storage must be allocated */
    /* separator: defines the chars that separate tokens */
    /* max_token_chars: max num of chars allowed in a token */
    /* max_num_tokens: max num of tokens
    *****/

    char *strptr = "";
    int len, num_tokens = 0;

    // MAKE SURE THE INPUT IS NOT A NULL STRING
    if (!strcmp(string, "")) return(0);

    // PULL OFF THE TOKENS
    while (strptr != NULL) {
        if (num_tokens == 0) { // GET FIRST TOKEN
            strptr = strtok(string, separator);
            if ((len = strlen(strptr)) > max_token_chars) {
                fprintf(stderr, "ERROR..get_tokens");
                fprintf(stderr, "\n first token had %d chars", len);
                fprintf(stderr, "\n max token chars: ");
                fprintf(stderr, "%d", max_token_chars);
                fprintf(stderr, "\n");
                return(0);
            } else {
                strcpy(token[num_tokens++], strptr);
            }
        } else { // OTHER TOKENS
            strptr = strtok(NULL, separator);
            if (strptr != NULL) {
                if (num_tokens > max_num_tokens) {
                    fprintf(stderr, "ERROR..get_tokens");
                    fprintf(stderr, "\n exceeded max num tokens: ");
                    fprintf(stderr, "%d", max_num_tokens);
                    fprintf(stderr, "\n");
                    return(0);
                } else {
                    if ((len = strlen(strptr)) > max_token_chars) {
                        fprintf(stderr, "ERROR .. get_tokens");
                        fprintf(stderr, "\n subs token had %d chars", len);
                        fprintf(stderr, "\n max token chars: ");
                        fprintf(stderr, "%d", max_token_chars);
                        fprintf(stderr, "\n");
                    } else {
                        strcpy(token[num_tokens++], strptr);
                    }
                }
            }
        }
    }

    return(num_tokens);
}

/* END OF GET_TOKENS */

```

## Appendix F - FORTRAN Code

```

C *****
C PROGRAM DFAD
C
C SUBROUTINES:TAPEIO_I,TAPEIO_C,C32T35,BIN2DEC_D,WGSUTM,
C BIN2DEC_A,INOUT,PTGEN,PLOT:PLOT/LIB,UTMWGS
C LINKED BY :link.dfad.9.com
C *****

C R INTERGER MATRIX HOLDING THE FIC NUMBER OF THE ROADS
C V BYTE MATRIC HOLDING THE SURFACE FEATURE HEIGHTS

COMMON/A/BUFF36(12,8),BUFFBITS(288)

INTEGER*4 SBUFF(9),ifile

CHARACTER*80 OUTFILE,INFILE,filename
byte null
character*1 anull,aspace
BYTE BUFF36,BUFFBITS
equivalence(null,anull)

C***** READ INPUT PARAMETERS *****8

aspace = ''
null = 0

write(6,('Enter in DFAD file to process (no ext) ==> ',S))
read(5,(a40))filename
idx = index(filename,aspace)-1
infilc = filename(1:idx)//'.lvc'//anull
outfile = filename(1:idx)//'.36bits'

print *,infile
print *,outfile

ifile = iopenr(infile)
OPEN (UNIT=3,FILE=OUTFILE,
* STATUS='UNKNOWN',
* FORM='FORMATTED')

IRCNT=1
ic = 0
itot = 0
100 CONTINUE
ICOUNT=0

num = iread (ifile,SBUFF,36)
ircnt = ircnt + 1
itot = itot + num
if (mod(ircnt,100).eq.0)then
print *,'.Rec.bytes read,total bytes ',ircnt,num,itot
endif
if(num.eq.0)go to 999
c if(ic.gt.10000)go to 999

CALL C32T36(SBUFF)

do ilp = 1,8
ic = ic + 1
write(3,111)(buffbits(ij),ij=(36*(ilp-1)+1),(36*ilp)),ic
111 format(36i1.5x,i15)
enddo

do ij=1,9
sbuff(ij) = 0
enddo
do ij=1,288
buffbits(ij) = 0
enddo

if(1.eq.1)go to 100

999 CONTINUE
CLOSE (UNIT=3)
STOP
END

PROGRAM DFAD

```

```

C
C SUBROUTINES:BIN2DEC_D,
C LINKED BY :link.aries
C *****
C
C This program will read DFAD format off of tape and processes
C data into a .tvg format
C

COMMON/A/BUFF36(12,8),BUFFBITS(288)
COMMON/D/ADATA(21600*30)

INTEGER*4 STATUS
INTEGER*4 IFEAID(1000,3),ISMCA(14,3)
INTEGER*4 XINTNUM,ICOUNT,WAGWACN,WAGWACC,WAGCELL
INTEGER*4 porient,length,pwidth
INTEGER*4 ldirect,lwidth
INTEGER*4 astruct,atree,aroorf

CHARACTER*80 OUTFILE,INFILE,filename
CHARACTER*1 DSI(648),ACC(2700),SPACE,anull

BYTE BITS36(36),null
BYTE BUFF36,BUFFBITS,ADATA

DIMENSION X(8000),Y(8000)

EQUIVALENCE (STAT,STATUS)
equivalence (null,anull)

DATA SPACE/' '

C***** READ INPUT PARAMETERS *****8

write(6,('Enter in file to process ==> ',S))
read(5,(a40))filename
idx = index(filename,space)-1
infile = filename(1:idx)//'.36bits'
outfile = filename(1:idx)//'.dfad'

C ***** OPEN INPUT AND OUTPUT FILES *****

OPEN (UNIT=2,FILE=INFILE,
* STATUS='UNKNOWN',
* FORM='FORMATTED')
OPEN (UNIT=9,FILE=OUTFILE,
* STATUS='unknown',
* FORM='FORMATTED')

do i=1,3
do j=1,1000
ifcaid(j,i) = 0
enddo
enddo

100 CONTINUE
IRCNT = 1
ICOUNT = 0
NPTS = 0
IREJECT = 0
IPHTMAX = 0
IHEAD = 0
IDX = 1
ZONE = 38

C-----
C- Process Manuscript header, Data Set Identification Record, Accuracy Record
C-----

c- Manuscript Data Set Header consists of *6* 36-bit words
c-----

icc = 1
do i=1,6
read(2,500)bits36
500 format(36i1)
do j=1,36
buffbits(icc) = bits36(j)
icc = icc + 1
enddo
enddo
DO 120 J=1,288

```

```

      ADATA(J)=BUFFBITS(J)
120 CONTINUE

      CALL BIN2DEC_D(1,6,XINTNUM)
      PRINT *,XINTNUM
C   IF XINTNUM = 63 ----> END OF ALL MANUSCRIPTS
      IF(XINTNUM.EQ.63)GO TO 900

      IMAN=IMAN+1
      WRITE(6,1500)IMAN
1500 FORMAT(1X,'MANUSCRIPT #',I5)

      CALL BIN2DEC_D(10,3,XINTNUM)
      ILEV=XINTNUM

      CALL BIN2DEC_D(13,14,XINTNUM)
      WAGWACN=XINTNUM

      CALL BIN2DEC_D(27,5,XINTNUM)
      WAGWACC=XINTNUM

      CALL BIN2DEC_D(32,5,XINTNUM)
      WAGCELL=XINTNUM

      CALL BIN2DEC_D(37,36,XINTNUM)
      ILAT10=XINTNUM

      CALL BIN2DEC_D(73,36,XINTNUM)
      ILON10=XINTNUM

      ALON=ILON10/10./3600.
      ALAT=ILAT10/10./3600.
C   IF (ZONE.GT.30)ALON=-ALON
C   WRITE(9,1605)ALAT,ALON
1605 FORMAT(1X,'LAT,LOX',2F12.4)

      CALL BIN2DEC_D(109,18,XINTNUM)
      ILATMX=XINTNUM+ILAT10

      CALL BIN2DEC_D(127,18,XINTNUM)
      ILONMX=XINTNUM+ILON10

      ALONMX=ILONMX/10./3600.
      ALATMX=ILATMX/10./3600.
C   IF (ZONE.GT.30)ALONMX=-ALONMX

      WRITE(6,1100)IMAN,ILEV,WAGWACN,WAGWACC,WAGCELL,ALON,ALAT,ALONMX,
      X,
      *ALATMX

C-----
C- DSI consists of 648 bytes = *144* 36-bit words
C-----
      print *, 'Reading DSI'
      DO I=1,144
        READ(2,500)BITS36
      ENDDO
C   WRITE(9,1200)DSI(4),(DSI(1),I=7,33),(DSI(1),I=60,64),
C   *(DSI(1),I=65,79),(DSI(1),I=88,98),(DSI(1),I=127,141),
C   *(DSI(1),I=145,149),(DSI(1),I=160,163)

C-----
C- ACC consists of 2700 bytes = *600* 36-bit words
C-----
      print *, 'Reading ACC'
      DO I=1,600
        READ(2,500)BIT36
      ENDDO
C   WRITE(9,1300)(ACC(I),I=4,7),(ACC(I),I=12,15),(ACC(I),I=20,23),
C   * (ACC(I),I=56,57)

      ITOPLIM=21600

C-----
C- PROCESSING FEATURES
C-----
C   WRITE(9,2900)
2900 FORMAT(' ***** FEATURES BEING PROCESSED')

      i36 = 0

300 CONTINUE
C-----
C- Read Feature data header - *2* 36-BIT WORDS

```

```

C-----
      do i=1,288
        buffbits(i) = 0
      enddo

      idx = 1
      ice = 1
      do i=1,2

C-----Checking on checksum word-----
        if(i36.eq.600)then
          read(2,500)bits36
          write(9,500)bits36
          read(2,500)bits36
          write(9,500)bits36
          i36 = 0
          write(9,*)'*****checksum*****'
        endif

C=====
        READ(2,500)bits36
        i36 = i36 + 1
        write(9,*)i36,i36
        do j=1,36
          buffbits(ice) = bits36(j)
          ice = ice + 1
        enddo
      enddo

      DO J=1,288
        ADATA(J)=BUFFBITS(J)
      enddo

C   write(6,500)(buffbits(ij),ij=1,72)

      IHEAD=IHEAD+1

      CALL BIN2DEC_D(IDX,14,XINTNUM)
      IFAC=XINTNUM

      CALL BIN2DEC_D(IDX+14,2,XINTNUM)
      IFEATP=XINTNUM
C ***** TEST FOR END OF MANUSCRIPT *****
      IF(IFEATP.EQ.3)GO TO 800
      ITEST=IFEATP+1

C   CALL BIN2DEC_D(IDX+16+1,10-1,XINTNUM)
      CALL BIN2DEC_D(IDX+16,10,XINTNUM)
      IPHT=XINTNUM*2
      IF (IPHT.GT.IPHTMAX)IPHTMAX=IPHT

      CALL BIN2DEC_D(IDX+26,10,XINTNUM)
      IFICN=XINTNUM

      CALL BIN2DEC_D(IDX+36,5,XINTNUM)
      ISMC=XINTNUM

C----- Point Feature Specifics -----
      if(ifeatp.eq.0)then
        call bin2dec_d(idx+36+5,6,xintnum)
        porient = xintnum
        call bin2dec_d(idx+36+11,7,xintnum)
        plength = xintnum * 2
        if (ifcn.ge.230.and.ifcn.le.239)plength = xintnum * 20
        call bin2dec_d(idx+36+18,7,xintnum)
        pwidth = xintnum * 2
        if (ifcn.ge.230.and.ifcn.le.239)pwidth = xintnum * 20
        CALL BIN2DEC_D(IDX+61,11,XINTNUM)
        N=XINTNUM
        WRITE(6,1400)IFAC,IFEATP,IPHT,IFICN,ISMC,N,
        * porient,plength,pwidth
      endif

      if(ifeatp.eq.1)then
        call bin2dec_d(idx+36+5,2,xintnum)
        ldirect = xintnum
        call bin2dec_d(idx+36+7,7,xintnum)
        lwidth = xintnum * 2
        call bin2dec_d(idx+36+14,14,xintnum)
        lblank = xintnum
        CALL BIN2DEC_D(IDX+59,13,XINTNUM)
        N=XINTNUM
        WRITE(6,1401)IFAC,IFEATP,IPHT,IFICN,ISMC,N,
        * ldirect,lwidth
      endif

      if(ifeatp.eq.2)then
        call bin2dec_d(idx+36+5,4,xintnum)

```

```

astruct = xintnum
call bin2dec_d(idx+36+9,4,xintnum)
atree = xintnum * 10
call bin2dec_d(idx+36+13,4,xintnum)
aroot = xintnum * 10
call bin2dec_d(idx+36+17,6,xintnum)
ablk = xintnum
CALL BIN2DEC_D(IDX+59,13,XINTNUM)
N=XINTNUM
WRITE(6,1402)IFAC,IFEATP,IPHT,IFCN,ISMC,N,
* astruct,atree,aroot
endif

IF (IFCN.GT.0 .AND. IFCN.LE.1000 .AND.
* ITEST.GT.0 .AND. ITEST.LE.3)
* IFEAID(IFCN,ITEST)=IFEAID(IFCN,ITEST)+1
IF (ISMC.GT.0 .AND. ISMC.LE.14 .AND.
* ITEST.GT.0 .AND. ITEST.LE.3)
* ISMCAR(ISMC,ITEST)=ISMCAR(ISMC,ITEST)+1
IF (ISMC.LE.0 .OR. ISMC.GT.14)IMISS=IMISS+1

350 CONTINUE

C-----
c- Record of data points n# of 36-bit words
C-----
idx = 1
icc = 1
c print *, 'Reading ',n,' 36-bit words'
do i=1,n
c-----Checking on checksum word-----
if(i36.eq.600)then
read(2,500)bits36
c write(9,500)bits36
read(2,500)bits36
c write(9,500)bits36
i36 = 0
c write(9,*)'*****Checksum*****'
endif
C-----
read(2,500)bits36
i36 = i36 + 1
c write(9,*)i36
do j=1,i36
adata(icc) = bits36(j)
icc = icc + 1
enddo
cnddo

NPTS=0
DO J=1,N
NPTS=NPTS+1
CALL BIN2DEC_D(IDX+1,18-1,XINTNUM)
c Y(NPTS)=XINTNUM/36000.
Y(NPTS)=XINTNUM/36000. + ALAT
c if(y(npts).eq.alat.and.npts.gt.5)y(npts) = alatmx
CALL BIN2DEC_D(IDX+18+1,18-1,XINTNUM)
c X(NPTS)=XINTNUM/36000.
X(NPTS)=XINTNUM/36000. + ALON
c if(x(npts).eq.alon.and.npts.gt.5)x(npts) = alonmx
c IF (ZONE.GT.30)X(NPTS)=XINTNUM/36000. + ALON
IDX=IDX + 36
ENDDO

write(9,1604)ifeatp

if (ifeatp.eq.0)then
WRITE(9,1600)IFAC,IFCN,ISMC,IPHT,
* poricent,plength,pwidth,NPTS
endif
if (ifeatp.eq.1)then
WRITE(9,1601)IFAC,IFCN,ISMC,IPHT,
* ldirect,lwidth,NPTS
endif
if (ifeatp.eq.2)then
WRITE(9,1602)IFAC,IFCN,ISMC,IPHT,
* astruct,atree,aroot,NPTS
endif

1600 format(8I5)
1601 format(7I5)
1602 format(8I5)
1604 format(I2)

```

```

DO MM=1,NPTS
WRITE(9,9876)X(MM),Y(MM)
9876 FORMAT(2F15.10)
ENDDO

GO TO 300

C *****
800 CONTINUE
c WRITE(9,3400)(IHEAD-1),IREJECT,IPHTMAX
3400 FORMAT(' END OF MANUSCRIPT',/,
* 'TOTAL FACS =',I9,/,
* 'REJECTED FACS =',I9,/,
* 'MAXIMUM FEATURE HEIGHT =',I9,/)

C--- GET NEXT MANUSCRIPT

c idiff = 600 - i36 + 1
c print *, 'Number of 36 read = ',i36
c print *, 'Reading to end of record = ',idiff
c do i=1,idiff
c read(2,500)bits36
c enddo

c print *, 'Reading checksum word'
c read(2,500)bits36

GO TO 100

C-----
C ALL MANUSCRIPTS PROCESSED
C-----

900 CONTINUE

DO I=1,1000
IF (IFEAID(I,1).GT.0 .OR. IFEAID(I,2).GT.0 .OR.
* IFEAID(I,3).GT.0)
* WRITE(6,2700)I,(IFEAID(I,J),J=1,3)
enddo
WRITE(6,1700)(I,(ISMCAR(I,J),J=1,3),I=1,14),IMISS

CLOSE (UNIT=2)
CLOSE (UNIT=3)
close (unit=9)
STOP

1100 FORMAT(/,' MANUSCRIPT NUMBER:',I5,/,
* 'LEVEL NUMBER :',I5,/,
* 'WAG(WAC) NUMBER :',I5,/,
* 'WAG(WAC) CELL :',I5,/,
* 'WAG CELL :',I5,/,
* 'SOUTHWEST LON/LAT:',2F12.2,/,
* 'NORTHEAST LON/LAT:',2F12.2,/)
1200 FORMAT(/,' DATA SET ID RECORD:',/,
* ' DATA CLASSIFICATION :',A1,/,
* ' SECURITY HANDLING :',A1,/,
* ' PRODUCT TYPE :',A1,/,
* ' MANUSCRIPT REF. NUMBER:',A1,/,
* ' DATA EDITION NUMBER :',A1,/,
* ' MATCH/MERGE VERSION :',A1,/,
* ' MAINTENANCE DATE(YMM):',A1,/,
* ' MATCH/MERGE DATE(YMM):',A1,/,
* ' PRODUCT SPEC. STOCK NO:',A1,/,
* ' AMMENDMENT/CHANGE NO:',A1,/,
* ' DATE(YMM) :',A1,/,
* ' HORIZONTAL DATUM CODE:',A1,/,
* ' COMPILATION DATE(YMM):',A1,/)

1300 FORMAT(' ACCURACY RECORD:',/,
* '4X,4A1,' = ABSOLUTE HORIZONTAL ACCURACY (M)',/,
* '4X,4A1,' = POINT-TO-POINT HORIZONTAL ACCURACY (M)',/,
* '4X,4A1,' = VERTICAL HEIGHTING ACCURACY (M)',/,
* '4X,2A1,' = MULTIPLE ACCURACY OUTLINE FLAG',/,
* '10X,' 00 = NO ACCURACY SUBREGIONS',/,
* '10X,' 02-09 = NUMBER OF ACCURACY SUBREGIONS',//)

1400 FORMAT(1x,'-----Point-----',/,
* 2X,'FAC=',I5,' Fea type=',I1,' HT=',I4,/,
* 2X,'FIC=',I4,' SMCC=',I2,' Num coor fea=',I5,/,
* 2X,'Ori=',I2,' Len =',I3,' Width =',I3,/,
1401 FORMAT(1x,'-----Linear-----',/,

```

```

* 2X,'FAC=' ,I5,' Fea type=' ,I1,' HT=' ,I4,
* 2X,'FIC=' ,I4,' SMCC=' ,I2,' Num coor fea=' ,I5,
* 2X,'Dir=' ,I1,' Width=' ,I3)
1402 FORMAT(1x,'-----Area-----',/,
* 2X,'FAC=' ,I5,' Fea type=' ,I1,' HT=' ,I4,
* 2X,'FIC=' ,I4,' SMCC=' ,I2,' Num coor fea=' ,I5,
* 2X,'Strut=' ,I2,' Tree=' ,I3,' Roof=' ,I3)

1700 FORMAT(' SOIL MATERIAL TYPE    POINT LINE AREA',/
* 1X,I5,' METAL    ',3I5,/,
* 1X,I5,' PART METAL ',3I5,/,
* 1X,I5,' STONE    ',3I5,/,
* 1X,I5,' COMPOSITION ',3I5,/,
* 1X,I5,' EARTHWORKS ',3I5,/,
* 1X,I5,' WATER    ',3I5,/,
* 1X,I5,' DESERT    ',3I5,/,
* 1X,I5,' ROCK      ',3I5,/,
* 1X,I5,' CONCRETE  ',3I5,/,
* 1X,I5,' SOIL      ',3I5,/,
* 1X,I5,' MARSH     ',3I5,/,
* 1X,I5,' TREES     ',3I5,/,
* 1X,I5,' SNOW      ',3I5,/,
* 1X,I5,' ASPHALT   ',3I5,/,
* ' NO CODE      ',I5)
1800 FORMAT(' NUMBER OF CELLS WITH VEG HTS=' ,I8)
2700 FORMAT(' FEATURE NUMBER:POINT LINE AREA',4I5)

```

END

```

*****
*
* SUBROUTINE C32T36(BUFF)
*
*****
C-
C- This subroutine converts nine 32 bit words into
C- eight 36 bit words.
C-
COMMON /A/ BUFF36(12,8),BUFFBITS(288)
BYTE BUFF36,BUFFBITS
INTEGER*4 BUFF(9),AWORK,AOUT

aout = 0
c m=31
DO 10 I = 1,288
J = (I-1)/32 + 1
K = I - 1
8 IF (K,LT,32) GO TO 9
K = K - 32
GO TO 8
9 CONTINUE
AWORK = BUFF(J)
c asave = buff(j)
KK = 31 - K
CALL MVBITS(AWORK,KK,1,AOUT,0)
BUFFBITS(I) = AOUT
c call mvbits(asave,m,1,aout,0)
c buttbits(i) = aout
c m=m-1
c if(m.eq.-1)m=31
aout = 0
10 CONTINUE
c write(9,100)
c write(9,101)buff
c write(9,110)
c write(9,111)buffbits
c write(9,120)
c write(9,121)buttbits
c100 format(1x,'-----buff-----')
c110 format(1x,'-----buffbits-----')
c120 format(1x,'-----BUTTBits-----')
c101 format(9i16)
c111 format(36i2)
c121 format(32i2)

RETURN
END

```

```

*****
*
* SUBROUTINE BIN2DEC_D(I1,I2,XINTNUM)
*
*****
C-
C- Converts to decimal (DELTA REFERENCE)
C- I1 = starting bit location

```

```

C- I2 = number of bits to be used
C- XINTNUM = NUMBER RETURNED
C-----

```

```

COMMON /A/ BUFF36(12,8),BUFFBITS(288),buttbits(288)
COMMON /D/ ADATA(21600*30)
BYTE BUFF36,BUFFBITS,ADATA,buttbits
INTEGER*4 I1,I2,I3,J,ICOUNT
INTEGER*4 XINTNUM

```

```

ICOUNT=-1
XINTNUM=0
IWORK=0
I3=I2+I1-1

DO J=I3,I1,-1
ICOUNT=ICOUNT+1
ITEMP=ADATA(J)
IF(ICOUNT,LT,32)CALL MVBITS(ITEMP,0,1,XINTNUM,ICOUNT)
END DO

RETURN
END

```



## Appendix G - PV-WAVE® Procedures

aries.com

```
common hdr, deg_to_rad, rad_to_deg, alpha, ecc_sq, $
    lle_origin, rad_lle_origin, xyz_origin_uvw, $
    gsmtx01, gsmtx02, gsmtx21, gsmtx22, $
    rad_lle_uvw_conv, uvw_offset, xyz_conv
```

```
pro make_hdrbin, hfile
```

```
ilun = 5
get_lun, ilun
openr, ilun, hfile
f = fstat(ilun)
h = bytarr(f.size)
readu, ilun, h
point_lun, ilun, 0
id9 = where(h eq 9)
id32 = where(h eq 32)
isz = strtrim(string( id32(0) - id9(2) - 1 ), 2)
nx = 0
ny = 0
fmt1 = '(19x,i'+isz+',1x,i'+isz+',2x,/)'
readf, ilun, ny, nx, format=fmt1
fmt2 = '(22x,i4,1x,f8.5,4x,i4,1x,f8.5)'
readf, ilun, sw_lat_d, sw_lat_m, sw_lon_d, sw_lon_m, format=fmt2
readf, ilun, ne_lat_d, ne_lat_m, ne_lon_d, ne_lon_m, format=fmt2
fmt3 = '(30x,f8.6,1x,f8.6)'
readf, ilun, row_scl, col_scl, format=fmt3
free_lun, ilun
```

```
olun = 5
get_lun, olun
openw, olun, hfile+'.new'
writeu, olun, nx, ny
writeu, olun, sw_lat_d, sw_lat_m, sw_lon_d, sw_lon_m
writeu, olun, ne_lat_d, ne_lat_m, ne_lon_d, ne_lon_m
writeu, olun, row_scl, col_scl
free_lun, olun
```

```
return
end
```

```
pro sct_values
```

@aries.com

```
rad_lle_origin = {, lat:0.0D, lon:0.0D, clv:0.0D }
```

```
deg_to_rad = 1.74532925199D-2
rad_to_deg = 57.2957795132D
alpha = 6378137.0D
ecc_sq = (2.0D - (1.0D / 298.257223563D)) * (1.0D / 298.257223563D)
```

```
rad_lle_origin.lat = lle_origin.lat * deg_to_rad
rad_lle_origin.lon = lle_origin.lon * deg_to_rad
```

```
sin_lle_origin_lat = sin(rad_lle_origin.lat)
cos_lle_origin_lat = cos(rad_lle_origin.lat)
sin_lle_origin_lon = sin(rad_lle_origin.lon)
cos_lle_origin_lon = cos(rad_lle_origin.lon)
```

```
gsmtx01 = -sin_lle_origin_lat
gsmtx02 = cos_lle_origin_lat
gsmtx21 = cos_lle_origin_lat
gsmtx22 = sin_lle_origin_lat
```

```
xyz_origin_uvw = lle2uvw(lle_origin)
```

```
return
```

```
end
```

```
pro tcs2binary
```

```
rtyp = 0
ncnt = 0
xpt = 0.0
ypt = 0.0
cr = string("15b")
xmin = 9999999.0
ymin = 9999999.0
xmax = -9999999.0
```

```
ymin = -9999999.0
types = lonarr(3)
rfac = 0
rfic = 0
rsmc = 0
rhgt = 0
rori = 0
rlen = 0
rwid = 0
rpts = 0
rdir = 0
rstruc = 0
rtree = 0
roof = 0
manu = 1
lastfac = -1
```

```
fmt0 = '(8i5)' & sz0 = 8*5+1
fmt1 = '(7i5)' & sz1 = 7*5+1
fmt2 = '(8i5)' & sz2 = 8*5+1
fmtdata = '(f13.3,f13.3,f13.3)' & szdata = 13*3+1
```

```
fname = ''
print, 'Change TCS Text to Binary'
print, '-----'
print, 'Enter in DFAD file to process - NO extension'
read, fname
```

```
openr, 1, fname + '.tcs'
openw, 2, fname + '.tcsbin'
```

```
fin = fstat(1)
fts = float(fin.size)
```

```
pcnt = 0.0
```

```
writeu, 2, xmin, ymin, xmax, ymax
```

```
while not(coff(1))do begin
```

```
    readf, 1, rtyp, format='(i2)'
    writeu, 2, rtyp
    pcnt = pcnt + 2+1
```

```
case rtyp of
```

```
0: begin
    readf, 1, rfac, rfic, rsmc, rhgt, rori, rlen, rwid, rpts, format=fmt0
    writeu, 2, rfac, rfic, rsmc, rhgt, rori, rlen, rwid, rpts
    pcnt = pcnt + sz0
end
```

```
1: begin
    readf, 1, rfac, rfic, rsmc, rhgt, rdir, rwid, rpts, format=fmt1
    writeu, 2, rfac, rfic, rsmc, rhgt, rdir, rwid, rpts
    pcnt = pcnt + sz1
end
```

```
2: begin
    readf, 1, rfac, rfic, rsmc, rhgt, rstruc, rtree, roof, rpts, format=fmt2
    writeu, 2, rfac, rfic, rsmc, rhgt, rstruc, rtree, roof, rpts
    pcnt = pcnt + sz2
end
```

```
endcase
```

```
txpts = fltarr(rpts)
typts = fltarr(rpts)
tzpts = fltarr(rpts)
zpt = 0.0
```

```
for i=0, rpts-1 do begin
    readf, 1, xpt, ypt, zpt, format=fmtdata
    txpts(i) = xpt
    typts(i) = ypt
    tzpts(i) = zpt
    pcnt = pcnt + szdata
end
```

```
types(rtyp) = types(rtyp)+(rtyp+1)
```

```
writeu, 2, txpts, typts, tzpts
```

```
txmin = min(txpts, max=txmax)
tymin = min(typts, max=tymax)
if txmin lt xmin then xmin = txmin
if txmax gt xmax then xmax = txmax
if tymin lt ymin then ymin = tymin
if tymax gt ymax then ymax = tymax
```

```
if rfac lt lastfac then manu=manu+1
lastfac = rfac
```

```

print,manu,rtyp,rfic,rpts,rhgt, $
((pcent/fts)*100.0),cr,format=('$,5i5,2x,f6.2,"%%"a')

```

```

end

```

```

print,'
print,'Number of point, linear, area features'
print,types(0),types(1)/2,types(2)/3

```

```

close,1

```

```

point_lun,2,0
writeu,2,xmin,ymin,xmax,ymax

```

```

close,2

```

```

end

```

```

-----
pro investigate

```

```

rtyp = 0
ncnt = 0
xpt = 0.0
ypt = 0.0
cr = string("15b")
xmin = 9999999.0
ymin = 9999999.0
xmax = -9999999.0
ymax = -9999999.0
types = lonarr(3)
rfac = 0
rfic = 0
rsmc = 0
rhgt = 0
rori = 0
rlen = 0
rwid = 0
rpts = 0
rdir = 0
rstruc = 0
rtrec = 0
rroof = 0
manu = 1
lastfac = -1
fname = ''

```

```

print,'Enter in DFAD file to investigate'
read,fname
openr,1,fname + '.dfadbin'
openw,2,fname + '.check'

```

```

fin = fstat(1)
fts = float(fin.size)

```

```

pcent = 0L
ckent = 0L
sbits = 750L

```

```

readu,1,xmin,ymin,xmax,ymax

```

```

while not(cof(1))do begin

```

```

    readu,1,rtyp
    pcent = pcent + 2
    sbits = sbits + 2
    ckent = ckent + 2

```

```

case rtyp of

```

```

0: begin
    readu,1,rfac,rfic,rsmc,rhgt,rori,rlen,rwid,rpts
    pcent = pcent + (8*2)
end

```

```

1: begin
    readu,1,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts
    pcent = pcent + (7*2)
end

```

```

2: begin
    readu,1,rfac,rfic,rsmc,rhgt,rstruc,rtrec,rroof,rpts
    pcent = pcent + (8*2)
end

```

```

endcase

```

```

txpts = fltarr(rpts)
typts = fltarr(rpts)
readu,1,txpts,typts

```

```

pcent = pcent + (rpts * 4 * 2)
sbits = sbits + rpts
ckent = ckent + rpts

```

```

if (ckent gt 600)then begin
    ckent = 0
    sbits = sbits + 2
endif

```

```

if (rtyp eq 0)then goto,skip

```

```

diffx = txpts(0:*)-txpts(1:*)
diffy = typts(0:*)-typts(1:*)
idx = where(diffx gt 1.0 or diffx lt -1.0,xcnt)
idy = where(diffy gt 1.0 or diffy lt -1.0,ycnt)

```

```

if (xcnt gt 0 or ycnt gt 0)then begin
    printf,2,'FAC,x,y,pts ',rfac,xcnt,ycnt,rpts,sbits
    printf,2,'idx,idy ',idx,idy
    if (xcnt gt 0)then begin
        printf,2,'Xdiff = ',diffx(idx)
        printf,2,txpts(idx-1),typts(idx-1)
        printf,2,'---'
        printf,2,txpts(idx ),typts(idx)
        printf,2,'---'
        printf,2,txpts(idx+1),typts(idx+1)
    endif
    if (ycnt gt 0)then printf,2,'Ydiff = ',diffy(idy)
    printf,2,'-----'
endif

```

```

skip:

```

```

print,rfac,rtyp,rfic,rpts, $
((float(pcent)/fts)*100.0),cr,format=('$,4i5,2x,f6.2,"%%"a')

```

```

endwhile

```

```

print,pcent
close,1
close,2

```

```

stop

```

```

end

```

```

-----
pro aries2binary

```

```

rtyp = 0
ncnt = 0
xpt = 0.0
ypt = 0.0
cr = string("15b")
xmin = 9999999.0
ymin = 9999999.0
xmax = -9999999.0
ymax = -9999999.0
types = lonarr(3)
rfac = 0
rfic = 0
rsmc = 0
rhgt = 0
rori = 0
rlen = 0
rwid = 0
rpts = 0
rdir = 0
rstruc = 0
rtrec = 0
rroof = 0
manu = 1
lastfac = -1

```

```

fmt0 = '8i5' & sz0 = 8*5+1
fmt1 = '7i5' & sz1 = 7*5+1
fmt2 = '8i5' & sz2 = 8*5+1
fmtdata = 'f15.10,f15.10' & szdata = 15*2+1

```

```

fname = ''
print,'Change DFAD Text to Binary'
print,'-----'
print,'Enter in DFAD file to process - NO extension'
read,fname

```

```

openr,1,fname + '.dfad'
openw,2,fname + '.dfadbin'

```

```

fin = fstat(1)
fts = float(fin.size)

```

```

pcnt = 0.0

writeu,2,xmin,ymin,xmax,ymax

while not(cof(1))do begin

    readf,1,rtyp,format='(i2)'
    writeu,2,rtyp
    pcnt = pcnt + 2+1

    case rtyp of

    0: begin
        readf,1,rfac,rfic,rsmc,rhgt,rori,rln,rwid,rpts,format=fmt0
        writeu,2,rfac,rfic,rsmc,rhgt,rori,rln,rwid,rpts
        pcnt = pcnt + sz0
        end
    1: begin
        readf,1,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts,format=fmt1
        writeu,2,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts
        pcnt = pcnt + sz1
        end
    2: begin
        readf,1,rfac,rfic,rsmc,rhgt,rstruc,rtrce,rroof,rpts,format=fmt2
        writeu,2,rfac,rfic,rsmc,rhgt,rstruc,rtrce,rroof,rpts
        pcnt = pcnt + sz2
        end

    endcase

    txpts = ftarr(rpts)
    typts = ftarr(rpts)

    for i=0,rpts-1 do begin
        readf,1,xpt,ypt,format=fmtdata
        txpts(i) = xpt
        typts(i) = ypt
        pcnt = pcnt + szdata
    end

    types(rtyp) = types(rtyp)+(rtyp+1)

    writeu,2,txpts,typts

    txmin = min(txpts,max=txmax)
    tymin = min(typts,max=tymax)
    if txmin lt xmin then xmin = txmin
    if txmax gt xmax then xmax = txmax
    if tymin lt ymin then ymin = tymin
    if tymax gt ymax then ymax = tyman

    if rfac lt lastfac then manu=manu+1
    lastfac = rfac

    print,rfac,rtyp,rfic,rpts,rhgt, S
    ((pcnt/fts)*100.0),cr,format='(S,5i5.2x,f6.2,"%%"a)'

end

print,''
print,'Number of point, linear, area features'
print,types(0),types(1)/2,types(2)/3

close,1

point_lun,2,0
writeu,2,xmin,ymin,xmax,ymax

close,2

end

```

```

;
; $Id: color_palette.pro,v 1.1 1991/05/22 16:53:19 jeffry Exp $
;
; pro cpal
;+
; NAME: COLOR_PALETTE
; PURPOSE: To display the numerical values associated with a color table
;-
; Find interval to be used on the table
int=1
inx=1
if (!d.n_colors gt 128) then inx=.5
;
; Save currently active window number
holdw = !d.window
holdp = !p.color

```

```

;
; Set up window
yboxes=fix(!d.n_colors/(8*int))
yvalue=yboxes*8*int
if (yvalue ne !d.n_colors) then yboxes=yboxes+1
ysize=yboxes*40*inx
window,free=1,xsize=320,ysize=ysize
;
; Calculate when to switch printing the label in the opposite color
!p.color=!d.n_colors-1
change=yvalue*3/4
;
; Loop through colors
x=0
y=0
for i=0,!d.n_colors-1,int do begin
    tv,replicate(i,40,(40*inx)),x,y
    if (i ge change) then !p.color=0
    xyouts,x+5,y+5,trim(string(i),2),/device
    x=x+40
    if (x ge 40*8) then begin
        x=0
        y=y+(40*inx)
    endif
endfor
;
; Set back to previously active window
if holdw ge 0 then wset,holdw
!p.color=holdp
;
return
end

```

```

pro showbinary

rtyp = 0
ncnt = 0
xpt = 0.0
ypt = 0.0
cr = string("15b")
xmin = 999999.0
ymin = 999999.0
xmax = -999999.0
ymax = -999999.0
types = lonarr(3)
rfac = 0
rfic = 0
rsmc = 0
rhgt = 0
rori = 0
rln = 0
rwid = 0
rpts = 0
rdir = 0
rstruc = 0
rtrce = 0
rroof = 0
manu = 1
lastfac = -1
ficsav = intarr(3,1000)
ficol = [0,0,2,6,0,7,16,23,8,3,30, 8,15, 4,27, 3]

```

```

fname = ''
print,'Enter in DFAD file to view'
read,fname
openr,1,fname + '.dfadbin'

cfac = 0
print,'Enter FAC to view or zero for all'
read,cfac

```

```

pcnt = 1

readu,1,xmin,ymin,xmax,ymax

device,pseudo_color=8
window,0,xsize=900,ysize=900

tck_color
plot,[xmin,xmin,xmax,xmax,xmin],[ymin,ymax,ymax,ymin,ymin],xstyle=1,ystyle=1,S
color=0,tickformat='(f6.3)',background=1

while not(cof(1))do begin

    readu,1,rtyp

    case rtyp of

```

```

0: begin
    readu,1, rfac,rfic,rsmc,rhgt,rori,rten,rwid,rpts
end
1: begin
    readu,1, rfac,rfic,rsmc,rhgt,rdir,rwid,rpts
end
2: begin
    readu,1, rfac,rfic,rsmc,rhgt,rstruc,rtree,rroof,rpts
end

endcase

txpts = flarr(rpts)
typts = flarr(rpts)
readu,1,txpts,typts

if rfac ne cfac and cfac ne 0 then goto,skipper

ficidx = rfic/100
if (ficidx eq 9) then ficidx = 10 + ((rfic/10)-(rfic/100*10))

if rfac eq 1 then goto,skipper
wid = rwid/2
len = rlen/2

if rtyp eq 0 and rfic ne 420 then oplot,txpts,typts,color=ficcol(ficidx),psym=1,symsize=0.5
if rtyp eq 0 and rfic gt 400 and rfic lt 500 then begin
    print,rwid,rlen
    oplot,[txpts-wid,txpts-wid,txpts+wid,txpts+wid,txpts-wid], S
    [typts-len,typts+len,typts+len,typts-len,typts-len], S
    color=ficcol(ficidx)
    stop
endif
if rtyp eq 1 then oplot,txpts,typts,color=ficcol(ficidx)
if rtyp eq 2 then polyfill,txpts,typts,color=ficcol(ficidx)
ficsav(rytp,rfic) = ficsav(rytp,rfic) + 1

if rfac eq cfac and cfac ne 0 then begin
    stop
endif

pcnt = pcnt + 1

if rfac lt lastfac then begin
    manu=manu+1
    print,manu
endif
lastfac = rfac

skipper:
end

print,pcnt
close,1

openw,2,fname + '.fic'

printf,2,format="(1x,\"FIC\",5x,\"Point\",9x,\"Linear\",8x,\"Area\")"
printf,2,format="(9x,\"-----\",9x,\"-----\",8x,\"-----\")"
formfic = "(1x,i3,5x,i4,10x,i4,10x,i4)"
for i=100,999 do begin
    if ficsav(0,i) ne 0 or ficsav(1,i) ne 0 or ficsav(2,i) ne 0 then begin
        printf,2,i,ficsav(0,i),ficsav(1,i),ficsav(2,i),format=formfic
    endif
end
sum0 = long(total(ficsav(0,*)))
sum1 = long(total(ficsav(1,*)))
sum2 = long(total(ficsav(2,*)))
gtot = sum0+sum1+sum2
printf,2,format="(1x,\"Totals\",2x,\"-----\",9x,\"-----\",8x,\"-----\")"
printf,2,sum0,sum1,sum2,format="(9x,i5,9x,i5,9x,i5)"
printf,2,gtot,format="(,\"Grand total =\",i6)"
close,2

end

-----
function lle2uvw,lle

@aries.com

lon_offset = 0.0D
cos_lat = 0.0D
sin_lat = 0.0D
re = 0.0D

; Decimal degrees to radians

rad_lle.lat = lle.lat * deg_to_rad
rad_lle.lon = lle.lon * deg_to_rad

; Convert geodetic to JointStars-geocentric

lon_offset = rad_lle.lon - rad_lle.origin.lon;
cos_lat = cos( rad_lle.lat );
sin_lat = sin( rad_lle.lat );

re = alpha / sqrt( 1.0 - (ecc_sq * sin_lat * sin_lat) )

uvw_conv.x = (re + lle.clv) * cos_lat * cos( lon_offset )
uvw_conv.y = (re + lle.clv) * cos_lat * sin( lon_offset )
uvw_conv.z = ( (re * (1.0 - ecc_sq)) + lle.clv ) * sin_lat

return,uvw_conv

end

-----
function uvw2tcs,uvw

@aries.com

; convert uvw to tcs

uvw_offset.x = uvw.x - xyz_origin_uvw.x
uvw_offset.z = uvw.z - xyz_origin_uvw.z

xyz_conv.x = uvw.y
xyz_conv.y = gsmtx01 * uvw_offset.x + gsmtx21 * uvw_offset.z
xyz_conv.z = gsmtx02 * uvw_offset.x + gsmtx22 * uvw_offset.z

return,xyz_conv

end

-----
pro find_dfad_elev

@aries.com
lle_origin = {,lat:0.0D, lon:0.0D, clv:0.0D }
lle = {,lat:0.0D, lon:0.0D, clv:0.0D }
xyz = {, x:0.0D, y:0.0D, z:0.0D }
rad_lle = {,lat:0.0D, lon:0.0D, clv:0.0D }
uvw_conv = {,x:0.0D, y:0.0D, z:0.0D }
uvw_offset = {,x:0.0D, y:0.0D, z:0.0D }
xyz_conv = {,x:0.0D, y:0.0D, z:0.0D }

aname = ''

print,'Enter in Aries file to find elevations for - NO extensions'
read,aname

openr,1,aname+'.dfad'
openw,4,aname+'.tcs'

olat = 29.000000D & olon = 46.166666D & oclv = 0.0D

print,'Enter in Origin of dataset (Lat/Lon/Elev)'
lle_origin.lat = olat
lle_origin.lon = olon
lle_origin.clv = oclv

;----Initialize variables for Lat/Lon -> TCS conversion-----
set_values

edir = ''
print,'Enter in associated elevation directory'
read,edir
fname = ''
print,'Enter in datafile name'
read,fname

hname = edir+'/'+hdr'
make_hdrbin,hname

openr,2,edir+'/'+fname
openr,3,edir+'/'+hdr.new'

print,'Reading header info.....'
xd = 0 & yd = 0
nx = 0 & ny = 0
sw_lat_d=0.0 & sw_lat_m=0.0 & sw_lon_d=0.0 & sw_lon_m=0.0

```

```

ne_lat_d=0.0 & ne_lat_m=0.0 & ne_lon_d=0.0 & ne_lon_m = 0.0
row_scl=0.0 & col_scl=0.0

```

```

readu,3,nx,ny
readu,3,sw_lat_d,sw_lat_m,sw_lon_d,sw_lon_m, $
    ne_lat_d,ne_lat_m,ne_lon_d,ne_lon_m, $
    row_scl,col_scl

```

```

sw_lon_sec = ( sw_lon_d*3600.0 + sw_lon_m*60.0)
sw_lat_sec = ( sw_lat_d*3600.0 + sw_lat_m*60.0)
ne_lon_sec = ( ne_lon_d*3600.0 + ne_lon_m*60.0)
ne_lat_sec = ( ne_lat_d*3600.0 + ne_lat_m*60.0)

```

```

print,'Creating elevation array NX by NY ',nx,ny
z = intarr(nx,ny)
print,'Reading elevation array.....'
readu,2,z

```

```

;-----Processing Text file-----'

```

```

rtyp = 0 & ncnt = 0 & xpt = 0.0 & ypt = 0.0
cr = string("15b)
xmin = 9999999.0 & ymin = xmin
xmax = -9999999.0 & ymax = xmax
types = lonarr(3)
rfac = 0 & rfic = 0 & rsmc = 0 & rhgt = 0 & rori = 0 & rlen = 0
rwid = 0 & rpts = 0 & rdir = 0 & rstruc = 0 & rtrec = 0 & rroof = 0
manu = 1 & lastfac = -1 & pcnt = 0.0

```

```

fint0 = '(8i5)' & sz0 = 8*5+1
fint1 = '(7i5)' & sz1 = 7*5+1
fint2 = '(8i5)' & sz2 = 8*5+1
fintdata = '(f15.10,f15.10)' & szdata = 15*2+1
fintout = '(f13.3,f13.3,f13.3)'
fin = fstat(1)
fts = float(fin.size)

```

```

while not(cof(1))do begin

```

```

    readf,1,rtyp,format='(i2)'
    printf,4,rtyp,format='(i2)'
    pcnt = pcnt + 2+1

```

```

case rtyp of

```

```

0: begin

```

```

    readf,1,rfac,rfic,rsmc,rhgt,rori,rdir,rwid,rpts,format=fint0
    printf,4,rfac,rfic,rsmc,rhgt,rori,rdir,rwid,rpts,format=fint0
    pcnt = pcnt + sz0
end

```

```

1: begin

```

```

    readf,1,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts,format=fint1
    printf,4,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts,format=fint1
    pcnt = pcnt + sz1
end

```

```

2: begin

```

```

    readf,1,rfac,rfic,rsmc,rhgt,rstruc,rtrec,rroof,rpts,format=fint2
    printf,4,rfac,rfic,rsmc,rhgt,rstruc,rtrec,rroof,rpts,format=fint2
    pcnt = pcnt + sz2
end

```

```

endcase

```

```

txpts = flarr(rpts)
typts = flarr(rpts)
tzpts = flarr(rpts)

```

```

for i=0,rpts-1 do begin

```

```

    readf,1,xpt,ypt,format=fintdata
    txpts(i) = xpt
    typts(i) = ypt
    xidx = fix(((xpt*3600.0) - sw_lon_sec) / col_scl)
    yidx = fix(((ypt*3600.0) - sw_lat_sec) / row_scl)

```

```

    if (xpt*3600.0) lt sw_lon_sec or $
        (ypt*3600.0) lt sw_lat_sec or $
        (xpt*3600.0) gt ne_lon_sec or $
        (ypt*3600.0) gt ne_lat_sec then begin
        tzpts(i) = z(0,0)
    endif else begin
        tzpts(i) = z(xidx,yidx)
    endelse
    pcnt = pcnt + szdata
end

```

```

for i=0,rpts-1 do begin
    lle.lon = txpts(i)

```

```

    lle.lat = typts(i)
    lle.elv = tzpts(i)
    xyz = lle2tcs(lle)
    txpts(i) = xyz.x
    typts(i) = xyz.y
    tzpts(i) = xyz.z
end

```

```

for i=0,rpts-1 do begin

```

```

    xpt = txpts(i)
    ypt = typts(i)
    zpt = tzpts(i)
    printf,4,xpt,ypt,zpt,format=fintout
end

```

```

print,rtyp,rfic,rpts,rhgt,((pcnt/fts)*100.0),cr, $
    format='($,4i5,2x,f6.2,"%",a)'

```

```

endwhile

```

```

print,'

```

```

close,1
close,2
close,3
close,4

```

```

end

```

```

function lle2tcs,lle

```

```

@aries.com

```

```

;-----
; Convert geodetic to JointStars-geocentric (uvw)
;-----

```

```

uvw = lle2uvw(lle)

```

```

;-----
; convert JointStars-geocentric(uvw) to tcs
;-----

```

```

xyz = uvw2tcs(uvw)

```

```

return,xyz

```

```

end

```

```

pro showtcs

```

```

rtyp = 0
ncnt = 0
xpt = 0.0
ypt = 0.0
cr = string("15b)
xmin = 9999999.0
ymin = 9999999.0
xmax = -9999999.0
ymax = -9999999.0
types = lonarr(3)
rfac = 0
rfic = 0
rsmc = 0
rhgt = 0
rori = 0
rlen = 0
rwid = 0
rpts = 0
rdir = 0
rstruc = 0
rtrec = 0
rroof = 0
manu = 1
lastfac = -1
ficsav = intarr(3,1000)
ficol = [0,0,2,6,6,7,16,23,8,3,30,8,15,4,27,3]

```

```

fname = ''
print,'Enter in TCS file to view'
read,fname
openr,1,fname + '.tcsbin'

```

```

cfac = 0

```

```

pcnt = 1

```

```

unzoom:

readu,1,xmin,ymin,xmax,ymax
zxmin = xmin
zxmax = xmax
zymin = ymin
zymax = ymax

zoomit:

device,pseudo_color=8
window,0,xsize=900,ysize=900

tek_color
plot,[zxmin,zxmin,zxmax,zxmax,zxmin], $
    [zymin,zymax,zymax,zymin,zymin], $
    xstyle=1, ystyle=1, color=0, tickformat='(f6.3)', background=1

print, zxmin,zymin,zxmax,zymax
print,'-----'
while not(cof(1))do begin

    readu,1,rtyp

    case rtyp of

    0: begin
        readu,1,rfac,rfic,rsmc,rhgt,rori,rcln,rwid,rpts
        end
    1: begin
        readu,1,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts
        end
    2: begin
        readu,1,rfac,rfic,rsmc,rhgt,rstruc,rtrec,rroof,rpts
        end

    endcase

    txpts = flarr(rpts)
    typts = flarr(rpts)
    tzpts = flarr(rpts)
    readu,1,txpts,typts,tzpts

    txmin = min(txpts,max=txmax)
    tymin = min(typts,max=tymax)
    if txmin lt zxmin or $
    tymin lt zymin or $
    txmax gt zxmax or $
    tymax gt zymax then goto,skipper

    wid = rwid/2
    len = rlen/2

    if rfac ne cfac and cfac ne 0 then goto,skipper

    ficidx = rfic/100
    if (ficidx eq 9)then ficidx = 10 + ((rfic/10)-(rfic/100*10))

    if rfac eq 1 then goto,skipper

    oplot,txpts,typts,color=ficcol(ficidx),psym=1,sysmz=0.5
    if rtyp eq 0then begin

        oplot,[txpts-wid,txpts-wid,txpts+wid,txpts+wid,txpts-wid], $
            [typts-len,typts+len,typts-len,typts-len], $
            color=ficcol(ficidx)
        endif
    if rtyp eq 1 then oplot,txpts,typts,color=ficcol(ficidx)
    if rtyp eq 2 then polyfill,txpts,typts,color=ficcol(ficidx)
    ficsav(rtyp,rfic) = ficsav(rtyp,rfic) + 1
    if rfac eq cfac and cfac ne 0 then begin
        stop
    endif

    pcnt = pcnt + 1

skipper:
end

print,pcnt

ans = ''
print,'Do you want to zoom ? y/n/all'
read,ans
if ans eq 'n' then goto,theend
if ans eq 'all' then begin
    point_lun,1,0

```

```

goto,unzoom
endif

print,'Select first point'
cursor,xpt1,ypt1,/data
print,xpt1,ypt1
wait,1
print,'Select second point'
cursor,xpt2,ypt2,/data
print,xpt2,ypt2
zxmin = min([xpt1,xpt2])
zymin = min([ypt1,ypt2])
zxmax = max([xpt1,xpt2])
zymax = max([ypt1,ypt2])

point_lun,1,0
readu,1,xmin,ymin,xmax,ymax
goto,zoomit

theend:

close,1

openw,2,fname + '.fic'

printf,2,format='(1x,"FIC",5x,"Point",9x,"Linear",8x,"Area")'
printf,2,format='(9x,"-----",9x,"-----",8x,"----")'
formfic = "(1x,i3,5x,i4,10x,i4,10x,i4)"
for i=100,999 do begin
    if ficsav(0,i) ne 0 or ficsav(1,i) ne 0 or ficsav(2,i) ne 0 then begin
        printf,2,i,ficsav(0,i),ficsav(1,i),ficsav(2,i),format=formfic
    endif
end
sum0 = long(total(ficsav(0,*)))
sum1 = long(total(ficsav(1,*)))
sum2 = long(total(ficsav(2,*)))
gtot = sum0+sum1+sum2
printf,2,format='(1x,"Totals",2x,"-----",9x,"-----",8x,"----")'
printf,2,sum0,sum1,sum2,format='(9x,i5,9x,i5,9x,i5)'
printf,2,gtot,format='(/,,"Grand total = ",i6)'
close,2

end

-----
pro dfad2tcs

@aries.com
lle_origin = {,lat:0.0D, lon:0.0D, elv:0.0D }
lle = {,lat:0.0D, lon:0.0D, elv:0.0D }
xyz = {, x:0.0D, y:0.0D, z:0.0D }
rad_lle = {,lat:0.0D, lon:0.0D, elv:0.0D}
uvw_conv = {,x:0.0D, y:0.0D, z:0.0D}
uvw_offset = {,x:0.0D, y:0.0D, z:0.0D}
xyz_conv = {,x:0.0D, y:0.0D, z:0.0D}

aname = ''

print,'Enter in Aries file to find elevations for - NO extensions'
read,aname

openr,1,aname+'.dfad'
openw,4,aname+'.tcs'

olat = 29.000000D & olon = 46.166666D & oclv = 0.0D

print,'Enter in Origin of dataset (Lat/Lon/Elev)'

lle_origin.lat = olat
lle_origin.lon = olon
lle_origin.elv = oclv

;---Initialize variables for LatLon -> TCS conversion-----
set_values

edir = ''
print,'Enter in associated elevation directory'
read,edir
fname = ''
print,'Enter in datafile name'
read,fname

hname = edir+ '/hdr'
make_hdrbin,hname

openr,2,edir+ '/'+fname
openr,3,edir+ '/hdr.new'

```

```

print,'Reading header info.....'
xd = 0 & yd = 0
nx = 0 & ny = 0
sw_lat_d=0.0 & sw_lat_m=0.0 & sw_lon_d=0.0 & sw_lon_m = 0.0
ne_lat_d=0.0 & ne_lat_m=0.0 & ne_lon_d=0.0 & ne_lon_m = 0.0
row_scl=0.0 & col_scl=0.0

readu,3,nx,ny
readu,3,sw_lat_d,sw_lat_m,sw_lon_d,sw_lon_m, $
    ne_lat_d,ne_lat_m,ne_lon_d,ne_lon_m, $
    row_scl,col_scl

sw_lon_sec = ( sw_lon_d*3600.0 + sw_lon_m*60.0)
sw_lat_sec = ( sw_lat_d*3600.0 + sw_lat_m*60.0)
ne_lon_sec = ( ne_lon_d*3600.0 + ne_lon_m*60.0)
ne_lat_sec = ( ne_lat_d*3600.0 + ne_lat_m*60.0)

print,'Creating elevation array NX by NY ',nx,ny
z = intarr(nx,ny)

print,'Reading elevation array.....'
readu,2,z

;-----Processing Text file-----'
rtyp = 0 & ncnt = 0 & xpt = 0.0 & ypt = 0.0
cr = string("15b)
xmin = 999999.0 & ymin = xmin
xmax = -999999.0 & ymax = xmax
types = lonarr(3)
rfac = 0 & rfc = 0 & rsmc = 0 & rhgt = 0 & rori = 0 & rlen = 0
rwid = 0 & rpts = 0 & rdir = 0 & rstruc = 0 & rtrec = 0 & rroof = 0
manu = 1 & lastfac = -1 & pcnt = 0.0

fmt0 = '(8i5)' & sz0 = 8*5+1
fmt1 = '(7i5)' & sz1 = 7*5+1
fmt2 = '(8i5)' & sz2 = 8*5+1
fmtdata = '(f15.10,f15.10)' & szdata = 15*2+1
fmtout = '(f13.3,f13.3,f13.3)'
fin = fstat(1)
fls = float(fin,size)

while not(cof(1))do begin

    readf,1,rtyp,format='(i2)'
    printf,4,rtyp,format='(i2)'
    pcnt = pcnt + 2+1

    case rtyp of

0: begin
    readf,1,rfac,rfic,rsmc,rhgt,rori,rlen,rwid,rpts,format=fmt0
    printf,4,rfac,rfic,rsmc,rhgt,rori,rlen,rwid,rpts,format=fmt0
    pcnt = pcnt + sz0
    end
1: begin
    readf,1,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts,format=fmt1
    printf,4,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts,format=fmt1
    pcnt = pcnt + sz1
    end
2: begin
    readf,1,rfac,rfic,rsmc,rhgt,rstruc,rtrec,rroof,rpts,format=fmt2
    printf,4,rfac,rfic,rsmc,rhgt,rstruc,rtrec,rroof,rpts,format=fmt2
    pcnt = pcnt + sz2
    end

endcase

txpts = fltarr(rpts)
typts = fltarr(rpts)
tzpts = fltarr(rpts)

for i=0,rpts-1 do begin
    readf,1,xpt,ypt,format=fmtdata
    txpts(i) = xpt
    typts(i) = ypt
    xidx = fix(((xpt*3600.0) - sw_lon_sec) / col_scl)
    yidx = fix(((ypt*3600.0) - sw_lat_sec) / row_scl)

    if (xpt*3600.0) lt sw_lon_sec or $
    (ypt*3600.0) lt sw_lat_sec or $
    (xpt*3600.0) gt ne_lon_sec or $
    (ypt*3600.0) gt ne_lat_sec then begin
        tzpts(i) = z(0,0)
    end
    endelse begin
        tzpts(i) = z(xidx,yidx)
    end
    endelse
    pcnt = pcnt + szdata
end

```

```

end

for i=0,rpts-1 do begin
    llc.lon = txpts(i)
    llc.lat = typts(i)
    llc.elv = tzpts(i)
    xyz = llc2ics(llc)
    txpts(i) = xyz.x
    typts(i) = xyz.y
    tzpts(i) = xyz.z
end

for i=0,rpts-1 do begin
    xpt = txpts(i)
    ypt = typts(i)
    zpt = tzpts(i)
    printf,4,xpt,ypt,zpt,format=fmtout
end

print,rtyp,rfic,rpts,rhgt,((pcnt/fls)*100.0),cr, $
    format='($,4i5,2x,f6.2,"%",a)'

endwhile

print,'
close,1
close,2
close,3
close,4

end

-----
pro showall

rtyp = 0
ncnt = 0
xpt = 0.0
ypt = 0.0
cr = string("15b)
xmin = 999999.0
ymin = 999999.0
xmax = -999999.0
ymax = -999999.0
types = lonarr(3)
rfac = 0
rfic = 0
rsmc = 0
rhgt = 0
rori = 0
rlen = 0
rwid = 0
rpts = 0
rdir = 0
rstruc = 0
rtrec = 0
rroof = 0
manu = 1
lastfac = -1
ficsav = intarr(3,1000)
ficol = [0,0,2,6,6,7,16,23,8,3,30,8,15,4,27,3]

fname = ['aries1','aries2','aries3','aries4','aries5']

device,pseudo_color=8
window,0,xsize=900,ysize=900

xmin = 43.000
ymin = 26.000
xmax = 49.000
ymax = 32.000

tek_color
plot,[xmin,xmin,xmax,xmax,xmin],[ymin,ymin,ymax,ymin,ymin],xstyle=1,ystyle=1,$
    color=0,tickformat='(f6.3)',background=1

for afile = 0.4 do begin

openr,1,fname(afile) + '.dfadbin'

cfac = 0

pcnt = 1

readu,1,xmin,ymin,xmax,ymin
print,fname(afile),xmin,ymin,xmax,ymin

```

```

while not(cof(1))do begin

  readu,1,rtyp

  case rtyp of

0: begin
  readu,1,rfac,rfic,rsmc,rhgt,rori,rln,rwid,rpts
  end
1: begin
  readu,1,rfac,rfic,rsmc,rhgt,rdir,rwid,rpts
  end
2: begin
  readu,1,rfac,rfic,rsmc,rhgt,rstruc,rtrec,rroof,rpts
  end

endcase

txpts = flarr(rpts)
typts = flarr(rpts)
readu,1,txpts,typts

if rfac ne cfac and cfac ne 0 then goto,skipper

ficidx = rfic/100
if (ficidx eq 9)then ficidx = 10 + ((rfic/10)-(rfic/100*10))

if rfac eq 1 then goto,skipper

if rtyp eq 0 then oplot,txpts,typts,color=ficcol(ficidx),psym=1,symsize=0.5
if rtyp eq 1 then oplot,txpts,typts,color=ficcol(ficidx)
if rtyp eq 2 then polyfill,txpts,typts,color=ficcol(ficidx)
ficsav(rtyp,rfic) = ficsav(rtyp,rfic) + 1

if rfac eq cfac and cfac ne 0 then begin
  stop
endif

pcent = pcent + 1

lastfac = rfac

skipper:
end

close,1

olun = 0
get _lun,olun
openw,olun,fname(afile)+'_fic'

printf,olun,format='(1x,"FIC",5x,"Point",9x,"Linear",8x,"Area")'
printf,olun,format='(9x,"-----",9x,"-----",8x,"-----")'
formfic = "(1x,i3,5x,i4,10x,i4,10x,i4)"
for i=100,999 do begin
  if ficsav(0,i) ne 0 or ficsav(1,i) ne 0 or ficsav(2,i) ne 0 then begin
    printf,olun,i,ficsav(0,i),ficsav(1,i),ficsav(2,i),format=formfic
  endif
end
sum0 = long(total(ficsav(0,*)))
sum1 = long(total(ficsav(1,*)))
sum2 = long(total(ficsav(2,*)))
gtot = sum0+sum1+sum2
printf,olun,format='(1x,"Totals",2x,"-----",9x,"-----",8x,"-----")'
printf,olun,sum0,sum1,sum2,format='(9x,i5,9x,i5,9x,i5)'
printf,olun,gtot,format='(1x,"Grand total = ",i6)'
free _lun,olun
ficsav(*,*) = 0

endfor

end

-----

```



## Appendix H - Glossary

2-D	2-dimensional
3-D	3-dimensional
ADS	advanced distributed simulation
AFATDS	Advanced Field Artillery Tactical Data System
AML	ARC/INFO® Macro Language
API	application programmer interface
ARC/INFO®	a workstation GIS software package by Environmental Systems Research Institute
ArcView®	a desktop GIS software package by Environmental Systems Research Institute
ARIES	Advanced Radar Imaging Emulation System developed by Lockheed Martin Tactical Defense Systems, Litchfield Park, Arizona
AUTOGRAPHICS®	a GIS software package by Lockheed Martin Tactical Defense Systems, Akron, Ohio
C	a coding system for programming scientific problems to be solved by a computer
C <sup>4</sup> ISR	command, control, communications, computers, intelligence, surveillance and reconnaissance
CD-ROM	compact disk that can hold a large quantity of computer data
DFAD	digital feature analysis data
DIS	distributed interactive simulation
DT&E	developmental test and evaluation
DTED	digital terrain elevation data
ETE	End-To-End
Excel	a spreadsheet application by Microsoft®
FID	feature identification
FORTTRAN	a coding system for programming scientific problems to be solved by a computer
GIS	geographic information system
GUI	graphical user interface
ICD	interface control document
ID	identification
IEEE	Institute of Electrical and Electronics Engineers
ERDAS IMAGINE®	a geographic imaging suite by ERDAS®, Incorporated
JADS	Joint Advanced Distributed Simulation, Albuquerque, New Mexico
Janus	interactive, computer-based simulation of combat operations
Joint STARS	Joint Surveillance Target Attack Radar System
JTF	joint test force or Joint Test Force, Albuquerque, New Mexico
LGSM	light ground station module
LMTDS	Lockheed Martin Tactical Defense Systems
MIL-PRF	military performance specification
MIL-STD	military standard

MTI	moving target indicator
NIMA	National Imagery and Mapping Agency
OT&E	operational test and evaluation
PDU	protocol data unit
PV-WAVE®	a visual data analysis software package by Visual Numerics, Incorporated
R2V™	a raster to vector conversion software package by Able Software Company
RPS	radar processor simulation developed by Northrop Grumman, Melbourne, Florida
RWS	remote workstation
SAR	synthetic aperture radar
SEDRIS	Synthetic Environment Data Representation & Interchange Specification
STARS	surveillance target attack radar system
SWA	Southwest Asia
T&E	test and evaluation
TAFSM	Tactical Army Fire Support Model
TCS	Topocentric Coordinate System
TEXCOM	U.S. Army Test and Experimentation Command
TRAC	U.S. Army Training and Doctrine Command (TRADOC) Analysis Center
TRADOC	U.S. Army Training and Doctrine Command
UTM	Universal Transverse Mercator
VPF	Vector Product Format
VSTARS	Virtual Surveillance Target Attack Radar System
WSMR	White Sands Missile Range, New Mexico
WWW	world wide web

## Units of Measure

deg or °  
GB  
MB

degree  
gigabyte  
megabyte